Developing Oracle EBS Custom Extensions – The Right Way

Douglas Manning
Johns Hopkins University – Applied Physics Laboratory

Keyur Pancholi
Johns Hopkins University – Applied Physics Laboratory

Introduction

This white paper outlines how a longtime Oracle EBS customer [The Johns Hopkins University Applied Physics Laboratory (JHU-APL)] has developed and deployed (the right way) a range of simple to complex custom extensions to our Oracle EBS application (using the delivered and supported Oracle application framework and tools) in a cost effective and supported manner.

Background

APL is the largest division of one of the world's premier research institutions, Johns Hopkins University. It is home to approximately 4000 scientists, engineers and other staff who work on over 400 programs sponsored by the United States (U.S.) Government, as well as other organizations located across the globe.

APL has been using Oracle applications products since 1994, and is currently in the process of migrating to Oracle EBS version R12. APL's existing portfolio of Oracle Applications EBS modules includes the following: Purchasing, Accounts Payable, Fixed Assets, General Ledger, Inventory, Accounts Receivable and Cash Management.

Business Case for Customization of Oracle EBS

Overview

Over the past twenty years as the functionality and sophistication of commercially available software has increased and the price of this same software has decreased, a majority of small, medium and large organizations have purchased and deployed COTS software to run their core back-office business systems and even their more industry specific business processes. The major driver for these organizations in pursuing the COTS solution was the ability to access and use state-of-the-art automation and automation processes, without having to incur the high cost (money, time and risk) of developing this software within their organization, as well as supporting and maintaining these custom self-developed software solutions. Even the U.S. Government (in the 1990's) caught on to the trend of implementing COTS software solutions through legislation and executive directives (the Clinger-Cohen Act of 1996, the President's Management Agenda of 2002 and the Department of Defense Quadrennial Defense Review), and now mandates the acquisition and purchase of COTS software (where appropriate) and has mandated that this is the preferred software solution in order to reduce the short-term development and long-term maintenance and ownership costs of software for the U.S. Government.

The major issue area that has always plagued COTS software solutions, is in order for COTS software products to be cost effective to produce and to procure, the associated functionality that is built into these software products must be somewhat generic so that the widest range of customers (businesses and organizations) can use some aspect of the COTS functionality. For example, major COTS software suppliers like Oracle need to design and build a purchasing module that can be used by any "Joe's Regional Inc.", as well as a global giant such as Home Depot, in order to sell their software solution to the broadest customer base possible. Because of this inherent condition, COTS software products may not possess the ability to capture all of the unique processes and associated data that businesses and organizations require, in order to exist in their specific markets and areas of expertise.

Customization Considerations and Circumstances

The first and most paramount consideration that an organization should heavily investigate and pursue in reference to customizing COTS software products (like the Oracle EBS), is simply to try to avoid it at all costs. The current view of experts and leading edge organizations in all areas of information technology (academia, government and the commercial world) is that re-engineering business processes to fit and align with COTS software solutions is vital in terms of reducing ownership costs while providing the latest functionality and technology that an organization relies on to stay competitive and to succeed.

For example, "NASA and Boeing successfully built complex systems based on COTS products because they didn't practice business as usual. They changed their business and engineering processes to make the best use of the available COTS products. And they recognized the need for sustained engineering and management effort, such as evaluating new product releases to determine system impact and actively monitoring emerging technology, to keep the products and the system current." [1]

Even Oracle Corporation, in their 2006 presentation titled "Oracle E-Business Suite Customers: 10 Things You Can Do Now To Prepare For Fusion Applications," stated that Oracle E-Business Suite customers should rethink their customization strategy by taking inventory of their current customizations, evaluating them (in terms of what they are, how they were built, current requirements, obsolescence and their business value to the organization) and also plan for engineering their EBS products for the future by configuring rather than customizing.

By customizing the Oracle EBS, an organization incurs a higher cost and risk of product ownership. Higher ownership and risk costs associated with customization include the following:

1. Increased costs due to hiring, training and maintaining a competent IT staff to design, develop, implement, maintain and support Oracle EBS customizations.
2. Increased costs of applying Oracle EBS patches and upgrading your Oracle EBS investment to a newer version since customizations need to be evaluated, tested, and possibly re-implemented in association with patch and upgrade activities.
3. Increased risk to your Oracle EBS investment due to in-house developed customizations that break existing Oracle product transaction and/or interface processes.
4. Increased risk to your Oracle EBS investment due to Oracle patches that (when they are applied to your Oracle EBS product) break in-house custom developed transaction and/or interface processes.
5. Increased risk to your organization of Oracle not supporting your Oracle EBS investment due to your implementation of non-supported customizations.

The above costs and risks are not trivial, and should be reflected upon very carefully and conservatively when considering customizing your Oracle EBS investment.

As previously mentioned, the goal of all businesses and organizations that rely primarily on COTS software to run their enterprise (like Oracle EBS) must be, where possible and feasible, to have their business processes and procedures in-line with, and according to, the generic processes of the COTS software they rely upon and use. The key element associated with this industry goal is the phrase, where possible and feasible. "In many cases there will be a few instances where business process re-engineering is not possible. For example, due to policy or law, it may be necessary to build or acquire needed reports, interfaces, conversions, and extensions. In these cases, adding to the product must be done under strong configuration control." [2]

If re-engineering a business process or procedure to fit a standard (and possibly rigid) COTS software model will not provide the data or information that an organization needs to function, or leads to critical data integrity issues, unacceptable processing time or renders the process too complex or burdensome to reasonably use, then the benefits of not tailoring or not customizing the process to fit organizational reality may not be worth the extra cost or risk of implementing the customization.

A balanced business case for customization of the Oracle EBS would include the following:

1. Re-engineer as many business processes and procedures of an organization as possible, in order to align with and accommodate the COTS software. However, do not re-engineer business processes to align with COTS software, where the change will render critical data processing unusable or greatly impact an organization's data integrity.
2. Only develop software customizations to objects (custom reports, custom processes or functional extensions, custom objects, modifying existing objects, etc.) where no equivalent COTS software process exists or the existing process associated with the standard COTS object is unusable (too complex or burdensome).
3. Only use Oracle certified third-party COTS software products as bolt-on's to Oracle EBS, or custom developed bolt-on software (which is coded to Oracle development standards), in conjunction with the Oracle EBS, where no equivalent Oracle EBS process exists or the delivered COTS process is unusable.

Oracle EBS Customization Types

Part of the process of developing Oracle EBS custom extensions the right way, is having a clear understanding of the types of Oracle customizations that are out there, and knowing when to use them. The Oracle EBS product provides three core approaches for performing customization and product tailoring (the right way) in a supported framework. These three supported customization methodologies are known as product configuration, personalization and extensibility.

"Configuration provides setup and administrative choices using native features of the product. Some configuration examples include:

- Profile Options
- User defined fields (Flex field)
- Function Security Setup
- Data Security Setup

Personalization enables you to declaratively tailor the User Interface (UI) look-and-feel, layout or visibility of page content to suit a business need or user preference. Some personalization examples would include:

- Tailor the order in which table columns are displayed
- Tailor a query result
- Tailor the color scheme of the UI
- Folder forms
- Forms Personalization
- Oracle Application Framework (OAF)

Extensibility is about extending the functionality of an application beyond what can be done through personalization. Some extensibility examples include:

- Add new function flows
- Extend or override existing business logic
- Using Oracle Forms Developer, Oracle JDeveloper and Oracle Workflow" [3]

Configuration and personalization customization methodologies almost always involve the use of delivered Oracle EBS product standard functionality to change or tailor the Oracle EBS to perform in a specific (non-vanilla) custom fashion. Configuration and personalization usually never require the development and integration of custom code or objects to perform non-standard actions, events or processes.

Extensibility customizations, on the other hand, almost always involve the design and development of custom code or objects to change or augment the delivered Oracle EBS product. Along with the above mentioned custom flows and Oracle Forms, extensibility would include the design and development of custom reporting objects (Oracle Reports, Oracle BI Publisher, OBIEE and Hyperion), as well as integrating outside 3rd party products and processes with the Oracle EBS.

Integrating certified 3rd party software solutions (bolt-ons) to Oracle EBS represents the high-end of extensibility customizations. "In cases where a particular COTS product does not provide the entire set of required functionality, a 'bolt-on' could be used. A bolt-on is not part of the COTS software product but is typically part of a suite of software that has been certified to work with the product to provide the necessary additional functionality. These suites of software are integrated to provide the full set of needed functionality." [4]

An explanation of extensibility customizations would not be complete without mentioning an extensibility customization practice that should be avoided at all costs, the direct modification of Oracle EBS object code. Examples of this ultra bad habit would include the following:

- Directly modifying or changing object elements in an Oracle delivered form (deleting or modifying buttons, fields, regions, tabs, menu items, etc.).
- Directly modifying or changing object elements in an Oracle delivered report (deleting or modifying buttons, fields, regions, tabs, menu items, etc.).
- Directly modifying or changing standard product object elements (code) in an Oracle EBS delivered trigger, function, and/or procedure.

Directly modifying or changing delivered Oracle EBS object code (such as the ways mentioned above) is extremely risky due to the high probability that future Oracle bug fixes, CPU's, RUP's and other product patches that are specifically associated with the object that was customized, may produce unwanted, unintended or incorrect results, or may not be able to be applied at all (thus risking current and future product functionality). Also, this type of product alteration may jeopardize (make null and void) your Oracle EBS product support from Oracle Corporation.

JHU-APL Oracle EBS Custom Extensions

Configuration

Oracle EBS configuration custom extensions are the first place to start when contemplating the customization of your Oracle EBS product. Configuration custom extensions are the easiest to perform and maintain and represent the least risky and least intrusive of all Oracle EBS product custom extension approaches, since they use standard Oracle product data fields and functionality to extend the base Oracle EBS product.

In fact, Oracle designed their EBS product to provide the capability to capture and store customer specific industry data, within the delivered product, so customers would not have to write and maintain custom developed code and functionality to process just a few proprietary data elements and processes.

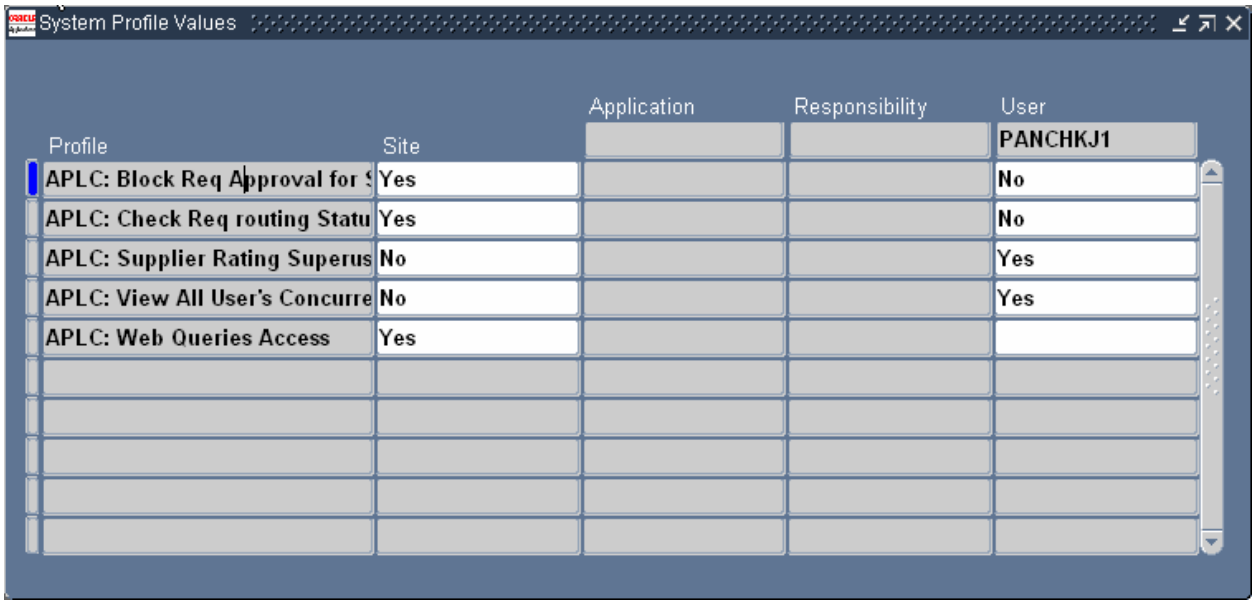
When configuration custom extensions are designed and deployed correctly, future Oracle bug fixes, CPU's, RUP's and other product patches that are later deployed within your Oracle EBS product should have no adverse effect on your custom extended configuration data or processes.

Some of the JHU-APL Oracle EBS configuration custom extensions that we have created and deployed (the right way) are detailed below (In some instances, proprietary data appearing on forms is intentionally blurred):

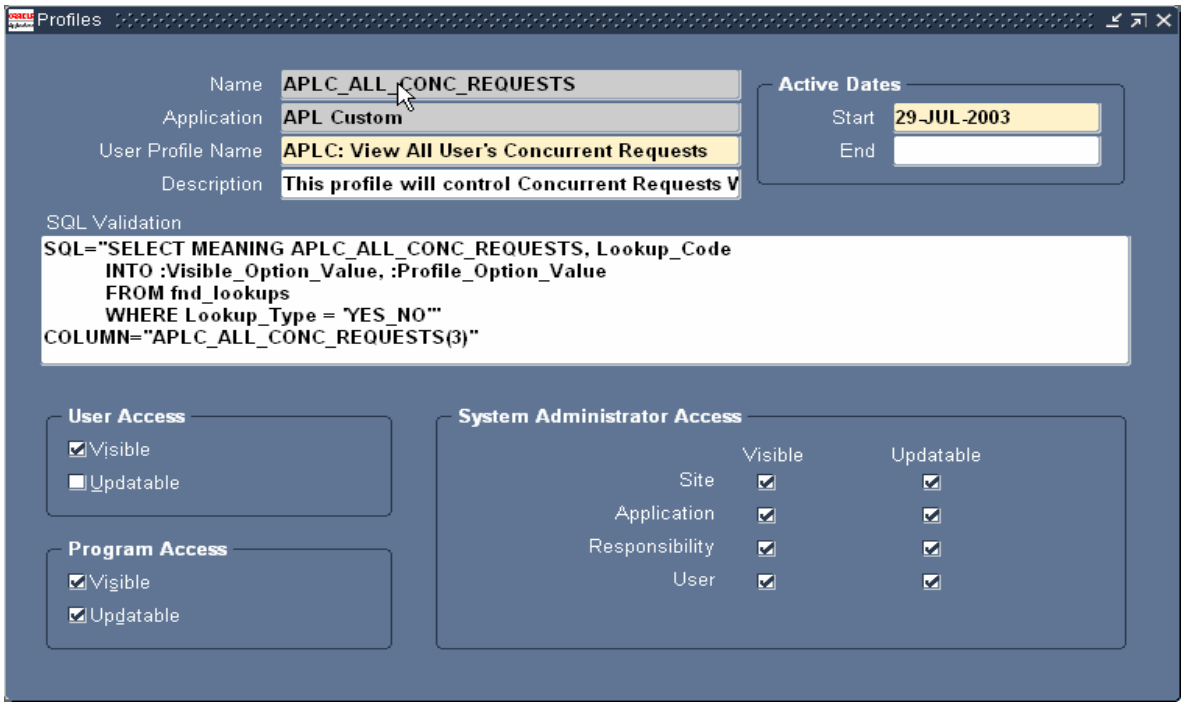
1. Profile Options

Profile Options are an Oracle EBS built-in feature (like a data flag), that provide a way to tailor and/or create custom processes within the standard Oracle EBS product. These custom processes can be setup at the site, application, responsibility and user levels.

An example of how to setup user profiles using the "System Profile Values" is listed below:

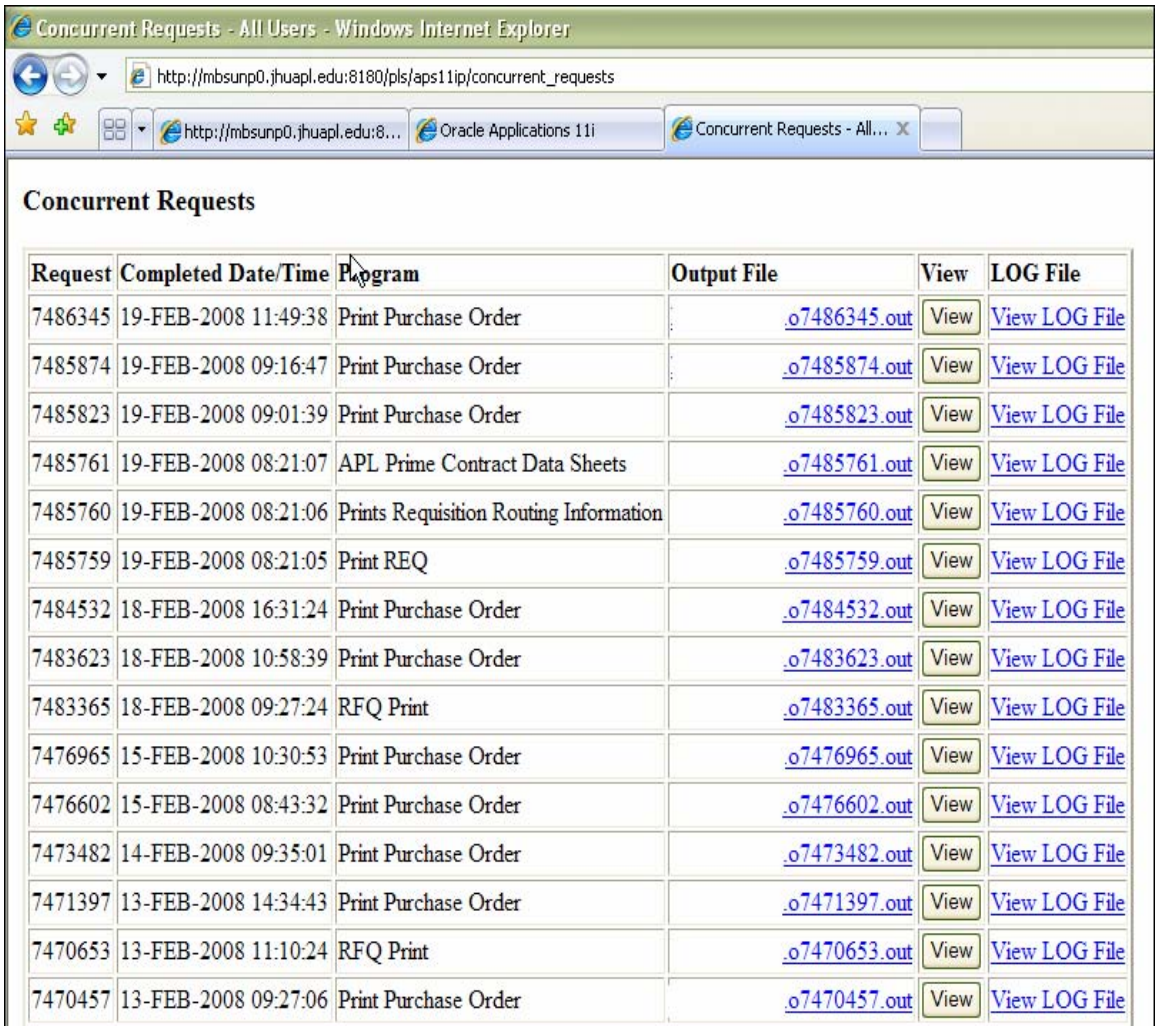


The associated custom SQL code and associated setup values for the profile option "View All User's Concurrent Requests" is detailed in the below Profile Options "Profiles" form. This profile option determines if a user can see another user's concurrent requests:



This profile option is very helpful for APL Oracle EBS support personnel because it gives them access to view the concurrent request outputs and/or concurrent request log file information of regular APL Oracle EBS users (who may be having trouble with the system), without having to use a 3rd party software product to takeover a user's workstation or without having to visit users to access more detailed information.

The below web form display's the results of a general user's concurrent request listing, associated with the option "View All User's Concurrent Requests" set to 'YES' for a specified APL Oracle EBS support user's account:



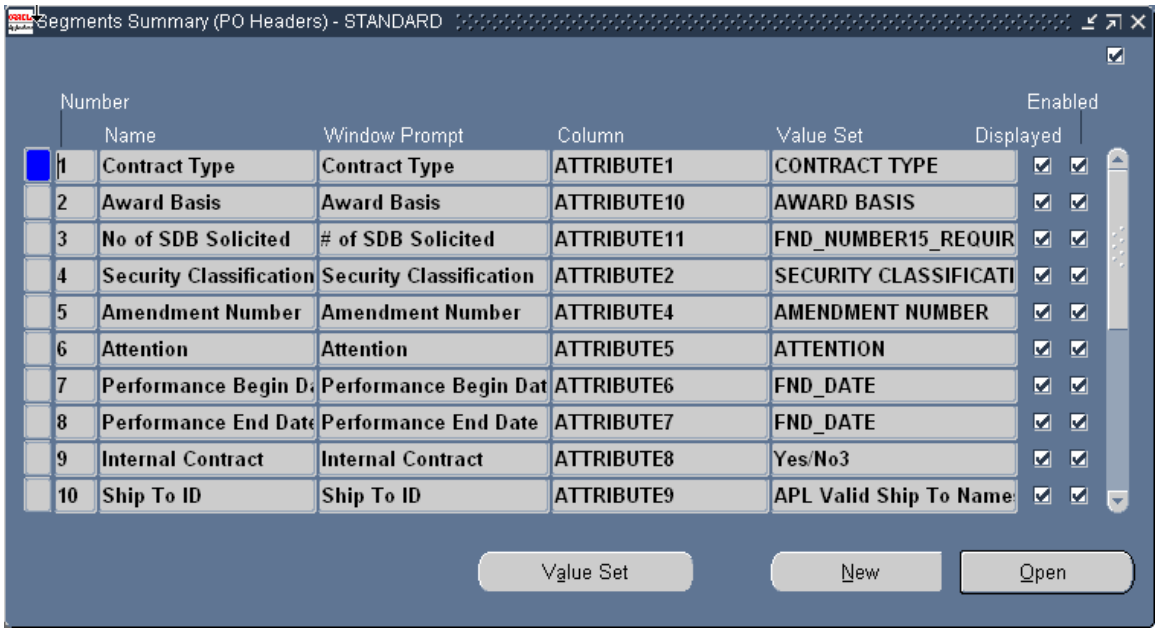
Concurrent Requests

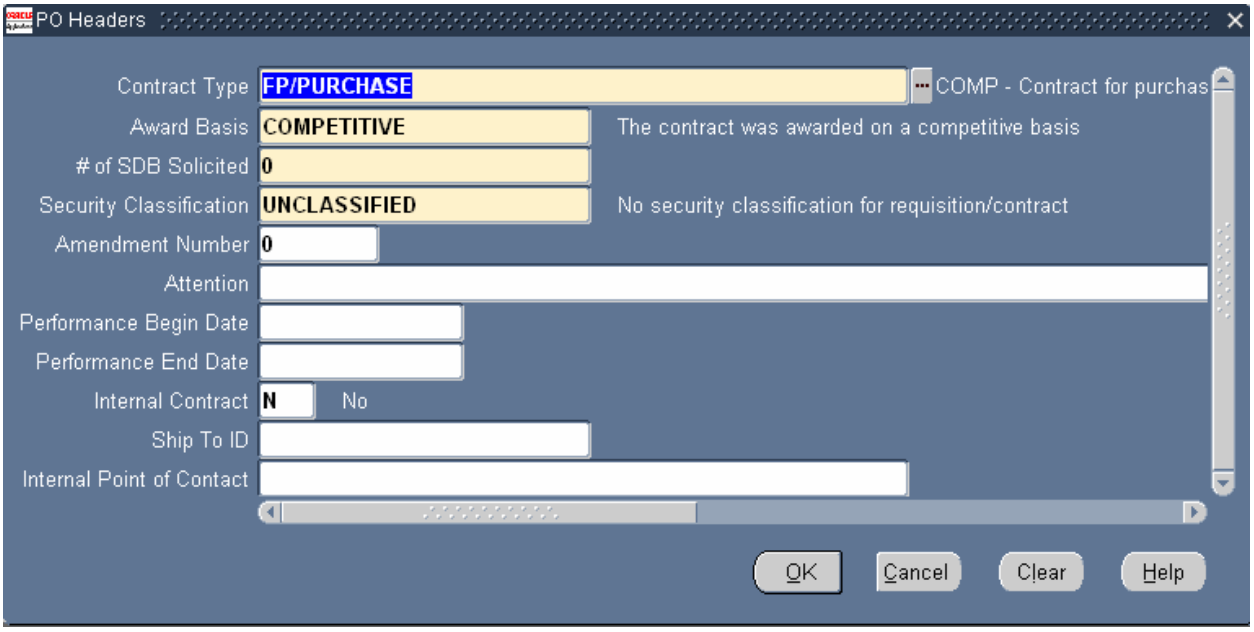
Request	Completed Date/Time	Program	Output File	View	LOG File
7486345	19-FEB-2008 11:49:38	Print Purchase Order	.o7486345.out	View	View LOG File
7485874	19-FEB-2008 09:16:47	Print Purchase Order	.o7485874.out	View	View LOG File
7485823	19-FEB-2008 09:01:39	Print Purchase Order	.o7485823.out	View	View LOG File
7485761	19-FEB-2008 08:21:07	APL Prime Contract Data Sheets	.o7485761.out	View	View LOG File
7485760	19-FEB-2008 08:21:06	Prints Requisition Routing Information	.o7485760.out	View	View LOG File
7485759	19-FEB-2008 08:21:05	Print REQ	.o7485759.out	View	View LOG File
7484532	18-FEB-2008 16:31:24	Print Purchase Order	.o7484532.out	View	View LOG File
7483623	18-FEB-2008 10:58:39	Print Purchase Order	.o7483623.out	View	View LOG File
7483365	18-FEB-2008 09:27:24	RFQ Print	.o7483365.out	View	View LOG File
7476965	15-FEB-2008 10:30:53	Print Purchase Order	.o7476965.out	View	View LOG File
7476602	15-FEB-2008 08:43:32	Print Purchase Order	.o7476602.out	View	View LOG File
7473482	14-FEB-2008 09:35:01	Print Purchase Order	.o7473482.out	View	View LOG File
7471397	13-FEB-2008 14:34:43	Print Purchase Order	.o7471397.out	View	View LOG File
7470653	13-FEB-2008 11:10:24	RFQ Print	.o7470653.out	View	View LOG File
7470457	13-FEB-2008 09:27:06	Print Purchase Order	.o7470457.out	View	View LOG File

2. Flex Fields

Flex fields are a mechanism and data component where the Oracle EBS allows customers to capture their specific industry or organization data (associated with a specific EBS forms transaction process) that are not contained in the standard EBS delivered functionality. These custom data elements are associated with a specific database table (e.g., PO.PO_HEADERS) and a specific database field within that table (e.g., ATTRIBUTE1). Most of the "Core" Oracle EBS module transaction forms (Enter Purchase Orders in the Purchasing module, Enter Invoices in the Accounts Payable module, etc.) provide the ability to create Flex fields.

JHU-APL takes full advantage of the Oracle EBS flex field data functionality, and uses quite a number of these data field elements for data capture and processing. The below figures display the JHU-APL Descriptive Flex field (DFF) structure and associated custom data fields that are used to process a JHU-APL Oracle Purchase Order (Enter Purchase Order form, PO Header zone):





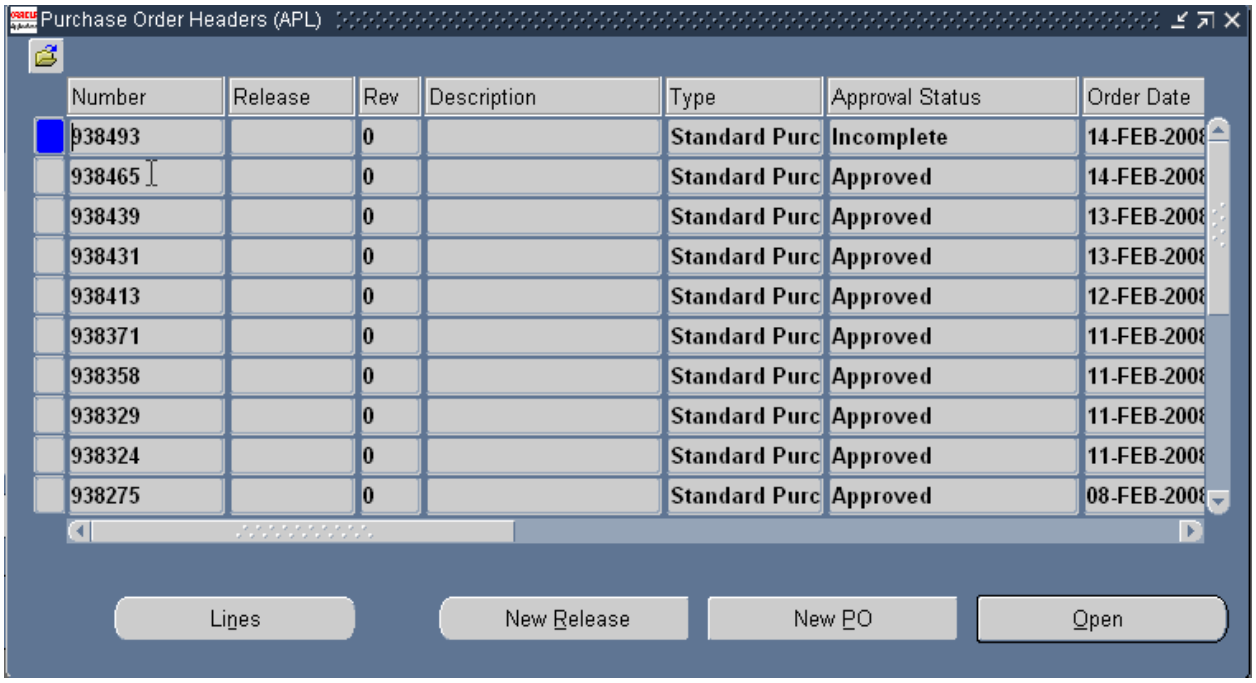
Notice the number of DFF's that JHU-APL needed to deploy in order to capture important organization data that was not part of the standard Oracle EBS "Enter Purchase Order" process. One of the major drawbacks of the Oracle EBS Flex Field custom extension functionality is actually the success of this functionality. Meaning, for a lot of Oracle EBS customers in certain Oracle module form areas, not enough DFF's exist, forcing Oracle EBS customers to look to other means to capture critical custom organization data.

With the above drawback being stated, Oracle EBS DFF's are an excellent way (and the right way) for an organization to extend the standard product functionality of Oracle EBS to capture important custom data.

3. Functional Data/Security Setup

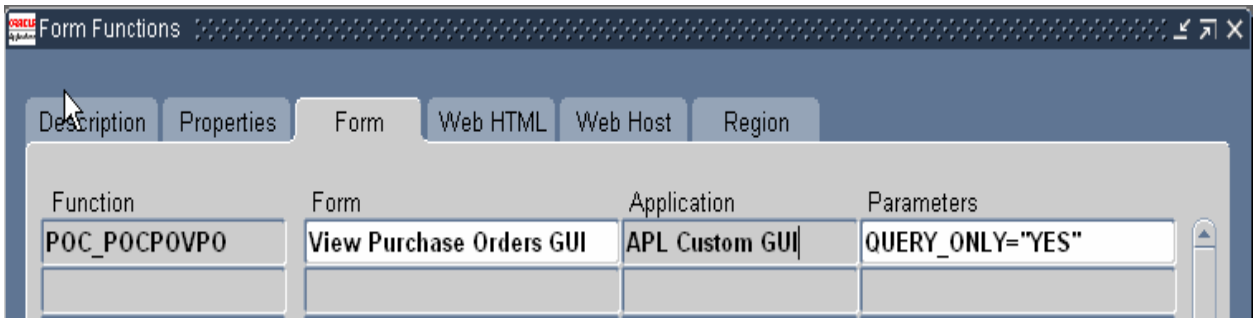
Oracle EBS functional data and security features are another way of custom configuring the Oracle EBS product without having to write code.

One of the drawbacks of version 11.5.7 of the Oracle EBS was that the purchasing product did not contain query-only purchasing functionality. The Purchasing summary form allowed users to not only query data but also allowed them to modify and change this data through the 'New Release' and 'New PO' buttons (see below form). APL wanted to provide our user base with a pure query-only capability in the purchasing area, in order to provide this detailed information to our community, without the ability to mistakenly change or modify data.



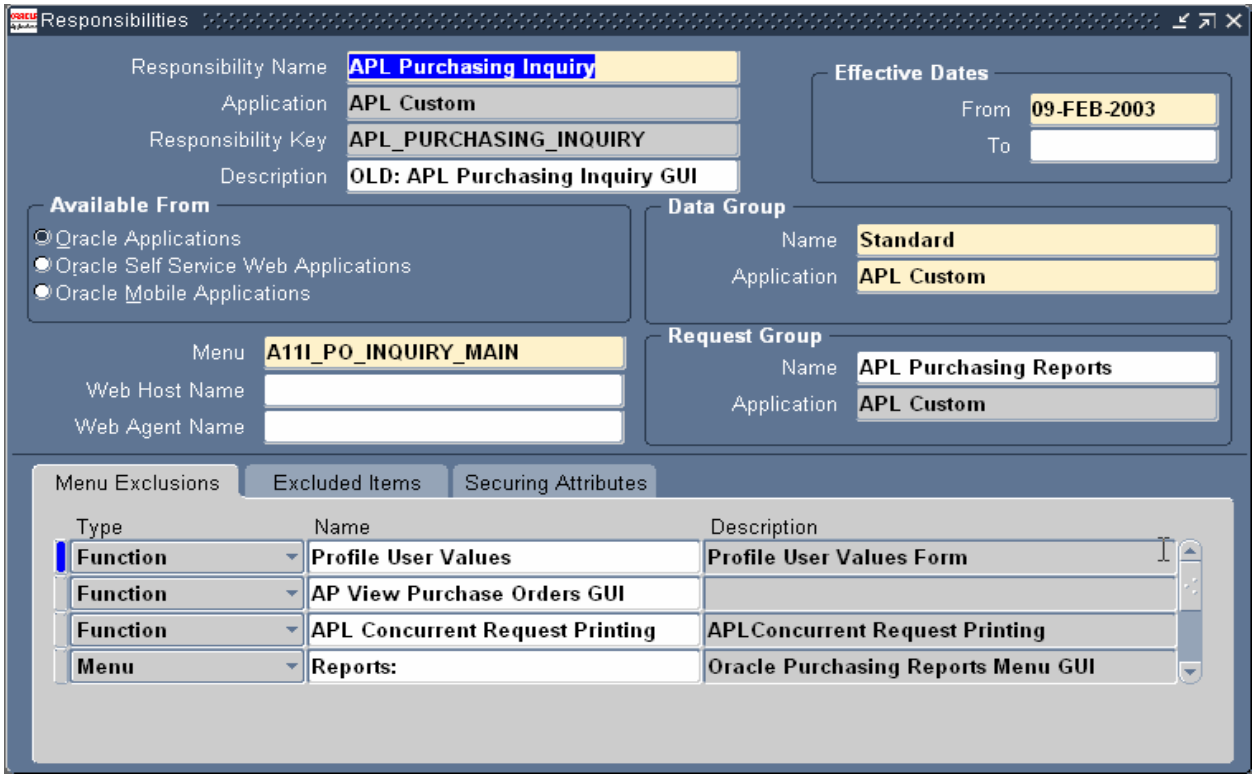
The way we provided this additional capability was through the use of Oracle's functional data/security setup features. The below screen shots detail how a "query-only" purchasing form and the associated "Purchasing Inquiry" responsibility were created.

The View Purchase Order form function was created using the default purchase order entry form, through the use of the parameter QUERY_ONLY. This function allows a user to access specific functionality in READ ONLY mode (see below form):



The associated APL "Purchasing Inquiry" responsibility and query-only purchasing form (see below forms), were created from a combination of the above custom form function and other relevant custom and delivered form functions:

APL Purchasing Inquiry Responsibility:



Query-only purchasing form:

Personalization

The second consideration when thinking about customizing your Oracle EBS product, from a visualization perspective, would be in the area of personalization. Personalization allows you to modify the look-and-feel and visualization components of standard Oracle EBS (UI) functionality, within a supported product frame work.

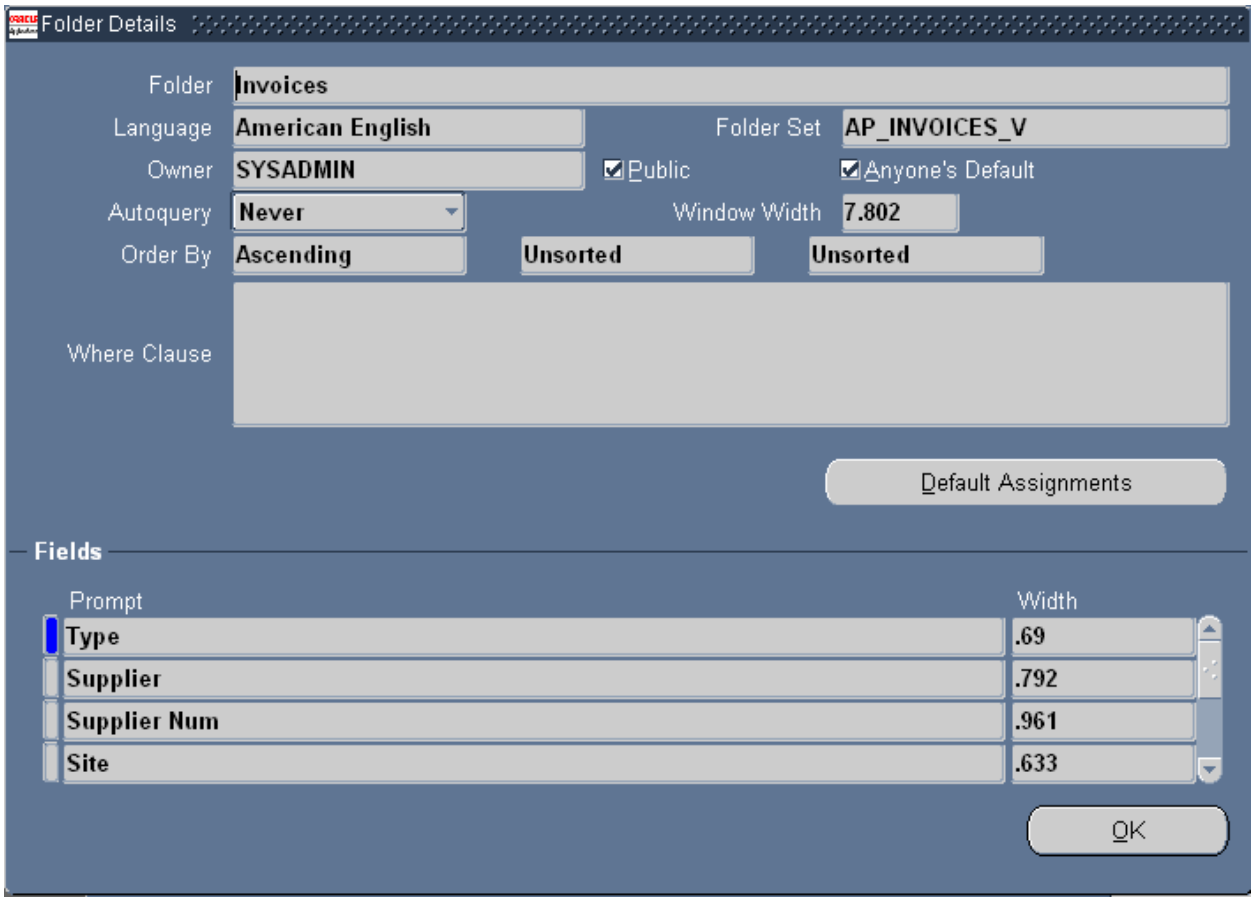
Personalization custom extensions are easy to perform and maintain. They are not as risky and intrusive as designing and developing custom Oracle forms or associated web forms for capturing, displaying and processing data, since this custom extension approach also uses standard Oracle product functionality to extend the base Oracle EBS product.

When personalization custom extensions are designed and deployed correctly, future Oracle bug fixes, CPU's, RUP's and other product patches that are later deployed within your Oracle EBS product should have no adverse effect on your custom extended personalization data or process.

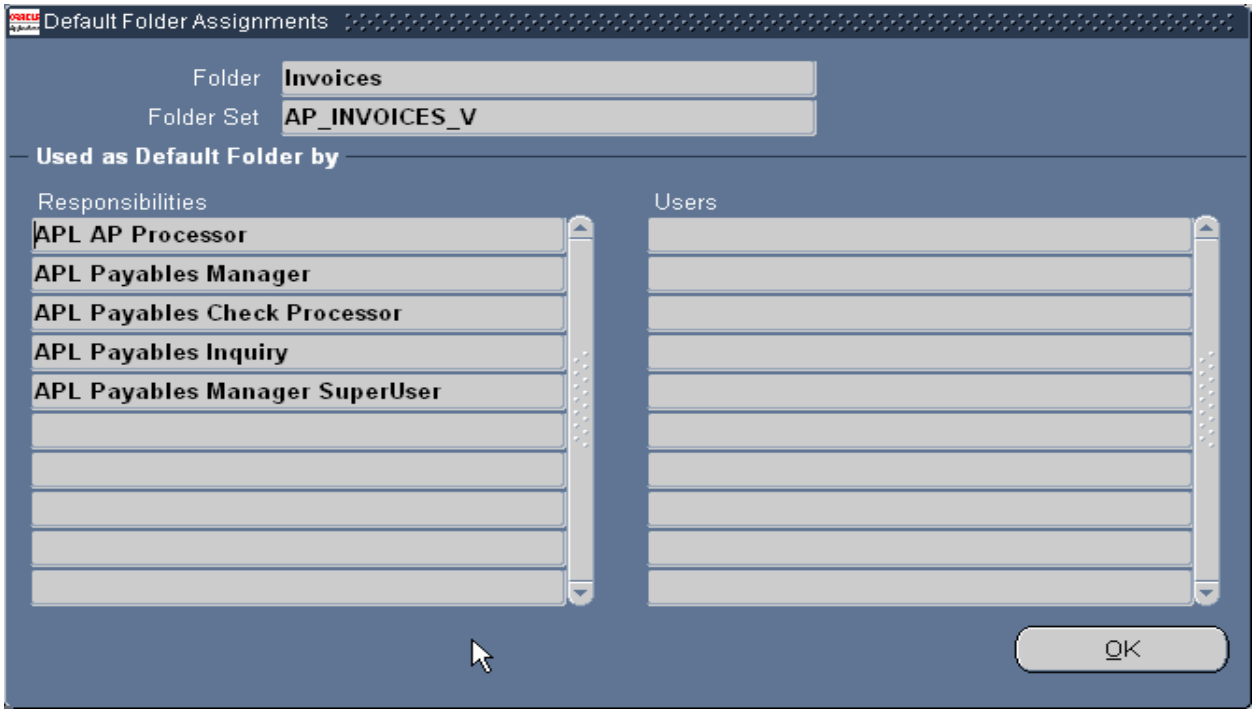
4. Query Result, Table Column Display Order and Folder Forms

One of the ways that JHU-APL was able to incorporate personalization custom extensions to our Oracle EBS product was through the use of Oracle built-in folder forms functionality. We changed the "default" AP Invoice form with the creation of a custom folder. Using this custom default folder, we are able to add fields, change the order of display of the fields and change the data sorting order. The below screen shots detail how this was performed:

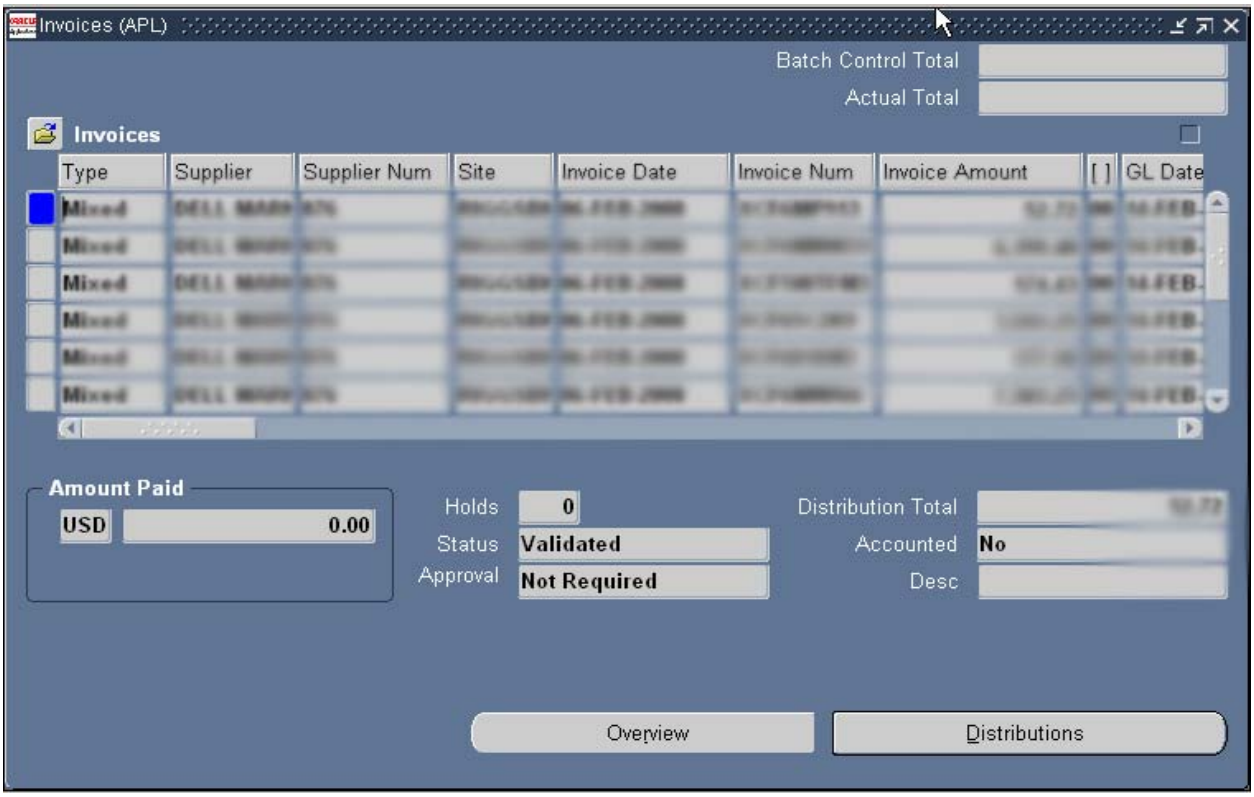
- First create a custom folder called "Invoices", with associated display fields and order of display:



- Second assign this custom folder to specified custom Oracle EBS responsibilities:



- Results of custom AP Invoice form folder (proprietary data intentionally blurred on form):



Extensibility

Extensibility is the Oracle EBS custom extension approach that is associated with designing, developing and maintaining custom code that is used in conjunction with the standard Oracle EBS delivered product.

Extensibility comes in many forms. It could be as small scale as a line or two of custom developed code incorporated into Oracle's CUSTOM.PLL forms library, or as big as numerous custom developed forms, reports and interfaces associated with an in-house developed process that is linked to the standard Oracle EBS product, or a full-blown 3rd party custom software product that contains data and processing interfaces to and from your Oracle EBS.

Extensibility custom extensions (depending on their context and associated requirements) can be tricky and will involve risk. And as noted in previous sections they are the most expensive (time, resources and money) custom extension option, however, they may be the only option an organization has, in order to perform a critical customer specific data and/or process function. Extensibility custom extensions should be used within your Oracle EBS only when the following cannot be met:

- Existing standard Oracle EBS data elements or processes cannot be used.
- Oracle EBS configuration custom extension(s) will not satisfy the requirement.
- Oracle EBS personalization custom extension(s) will not satisfy the requirement.

Some of the JHU-APL Oracle EBS extensibility custom extensions that we have created and deployed are detailed below:

5. New Function Flows

An example of how APL incorporated a new function and associated process flow to the Oracle EBS would include our custom function called "Local Printing". The "Local Printing" function was established to give our user community some flexibility in choosing a printer to send output to, since this custom process is not tied to a specific defaulted and assigned Oracle EBS user account printer or a printer within the Oracle EBS defined structure.

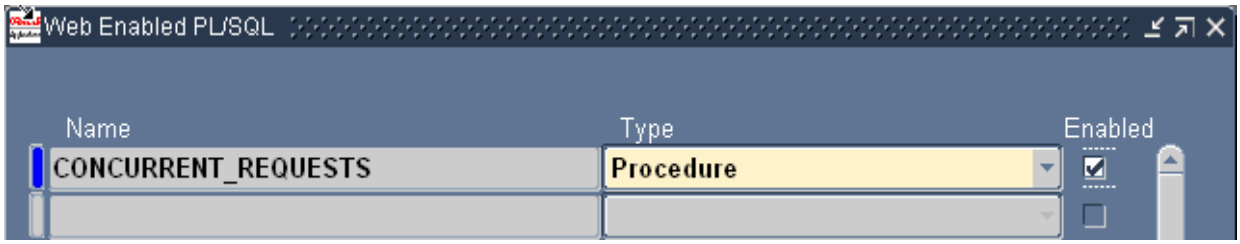
The APL "Local Printing" uses MOD PL/SQL and the web to access APL's printer universe, thus allowing a user to send their output, easily, to any APL network printer. The big functionality gain for APL is flexibility to the user community and less printer assignment and print queue maintenance for APL developers and support staff.

The MOD PL/SQL code for performing this functionality is detailed below:

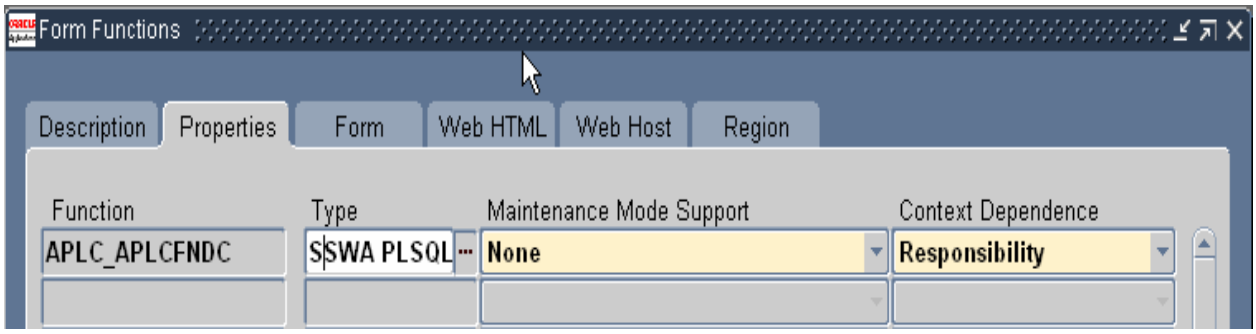
```
BEGIN

   FND_PROFILE.GET(NAME => 'APLC_ALL_CONC_REQUESTS', VAL => V_PROFILE_VAL);
   V_PROFILE_VAL := NVL(V_PROFILE_VAL, 'N');

   IF V_PROFILE_VAL = 'Y' --    in_user_name IS NULL
   --AND action = 'ALLUSERS'
   THEN
      HTP.HTMLOPEN;
      HTP.HEADOPEN;
      HTP.TITLE('Concurrent Reuqests - All Users');
      HTP.HEADCLOSE;
      HTP.BODYOPEN(CATTRIBUTES => ' BGCOLOR="#ffffff"  link="ffffff" vlink="ffffff"');
```

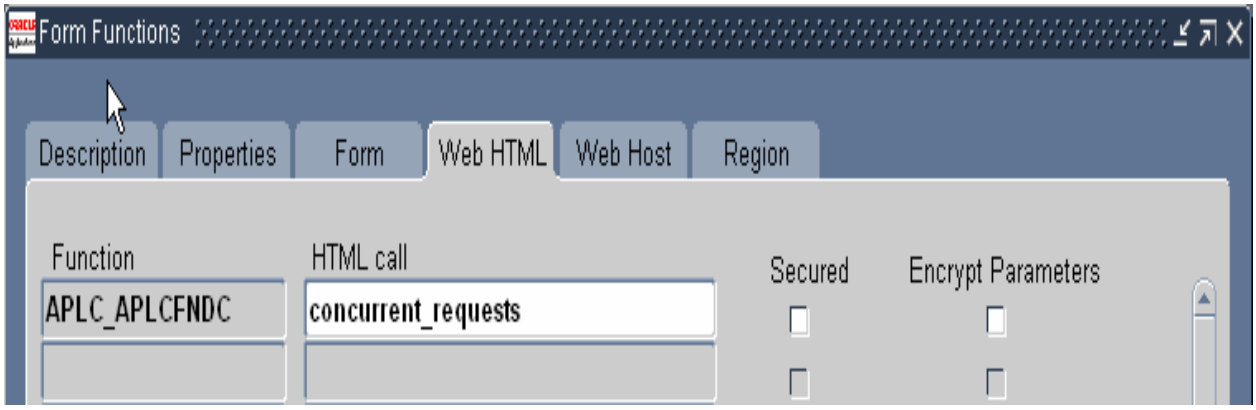
The associated Oracle EBS procedures and functions that need to be setup are listed below:

One - This procedure must be added to web enabled PL/SQL so that it can be used as a PL/SQL cartridge from the web server.

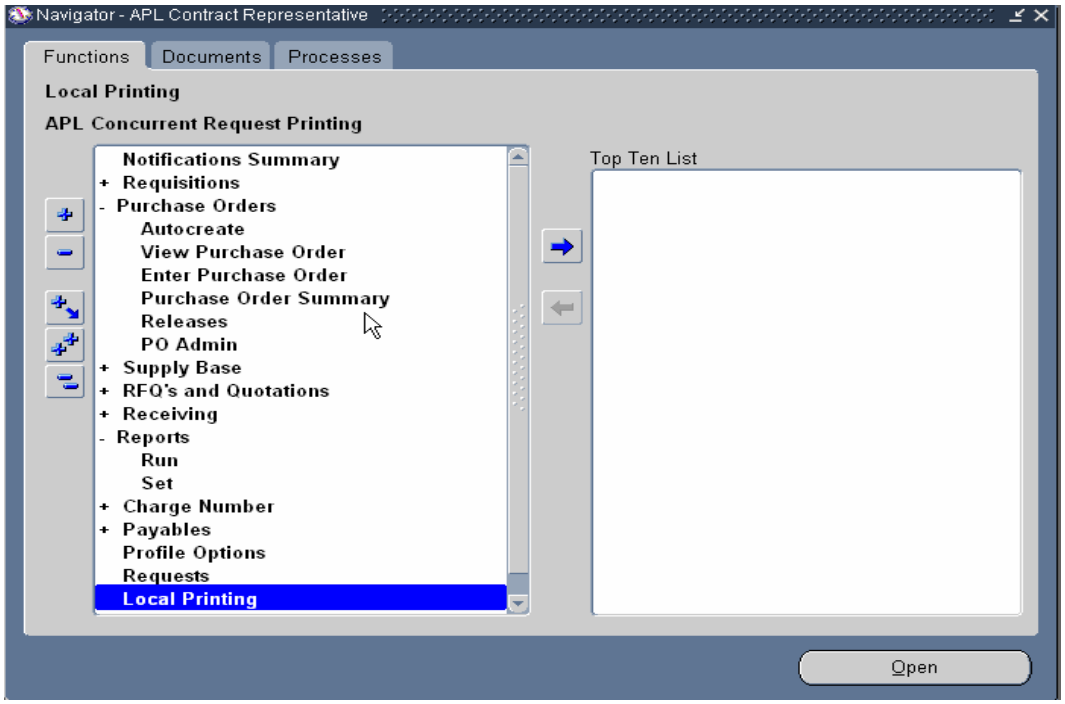


Two - A form function needs to be created in order to use it in the menu-tree

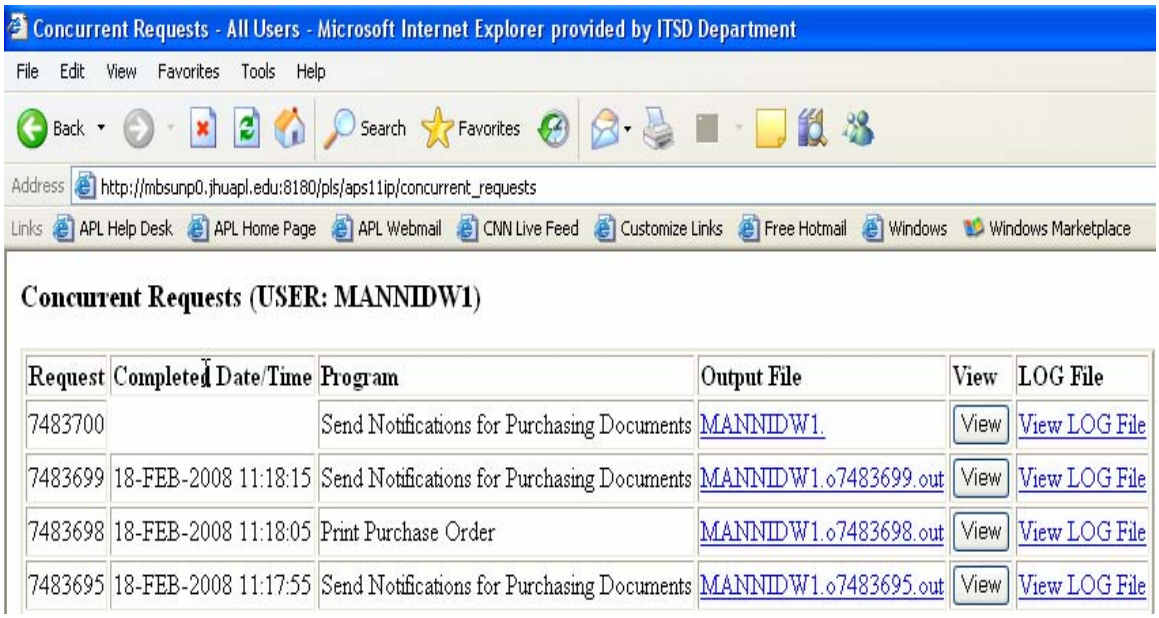




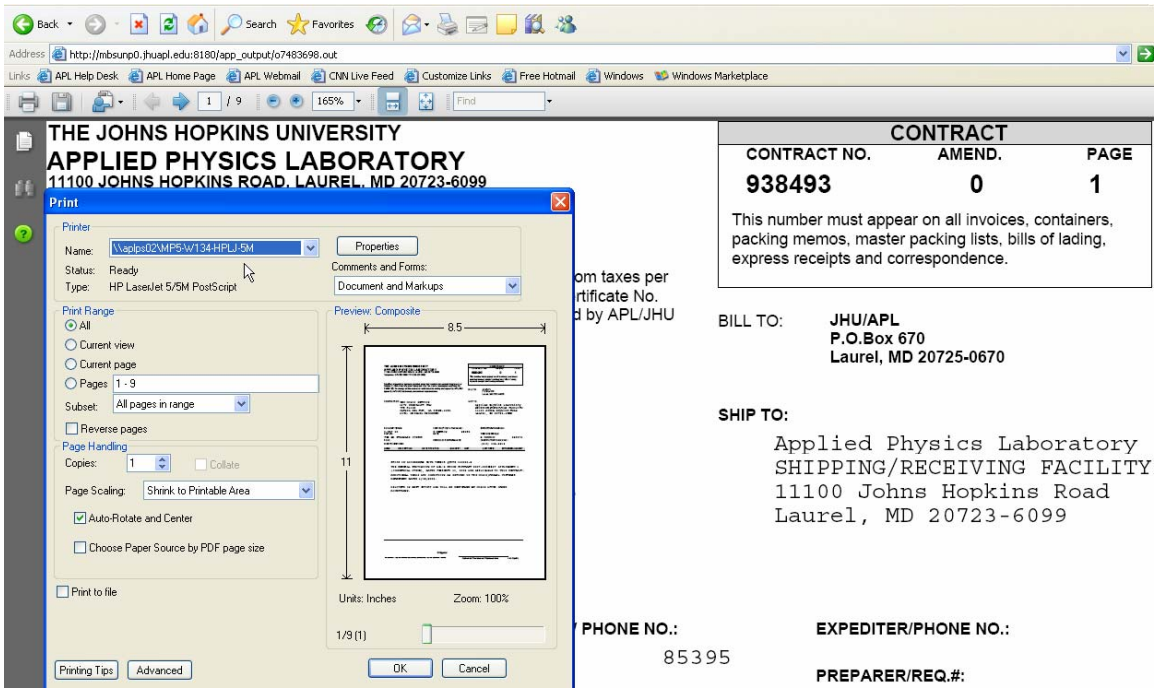
Three – After the function was setup and assigned to an APL custom Oracle EBS responsibility, a user would select the option “Local Printing” from this responsibility after running a report:



Four – The user would receive their concurrent request output, via the web, and then select their associated report from the output file list:



Five – Once the report was selected, a user could simply send their report to any APL network printer and not be restricted to their default printer or any other printer that needed to be maintained in the Oracle EBS:

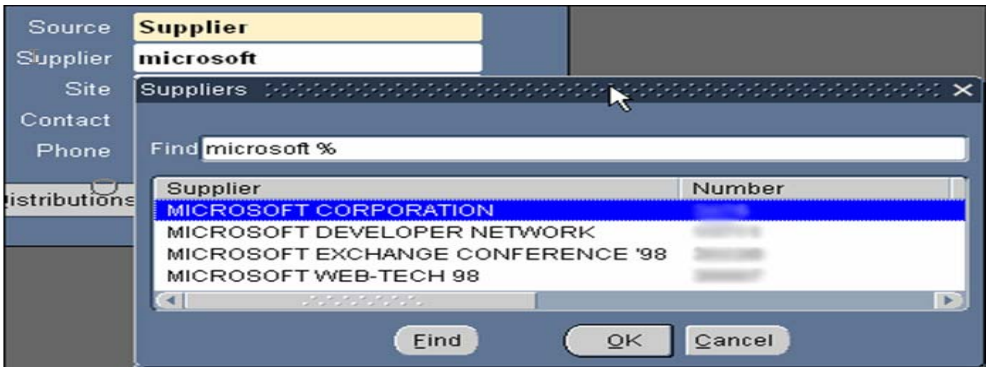


6. Override, Supersede and Extend Existing Business Logic

One of the primary and most effective ways of overriding and extending your Oracle EBS product, in a supported manner, is by using the CUSTOM.pll forms library. This built-in Oracle EBS functionality and process allows customers to incorporate custom (user exit and execution) code from within the Oracle EBS product to modify and change standard product forms functionality and characteristics in a safe and supported manner.

Over the years APL has incorporated dozens of CUSTOM.pll forms extensions to tailor standard Oracle EBS forms and transaction processes in a way that reduced support and data integrity and more effectively met our business process. The following are examples of APL Custom.pll extensions and associated library code:

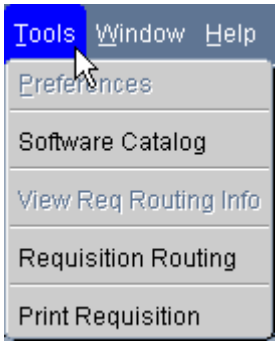
In the Oracle EBS "Enter Requisitions" form, we created our own supplier and supplier site validation process that forces users to select this information from a supplier pick-list. We implemented this change in order to capture better supplier data:



The associated enter requisition supplier entry field code contains the following elements:

```
IF In_Event = 'WHEN-NEW-FORM-INSTANCE'
THEN
   --
   IF in_form = 'POXRQERQ'
   THEN
      --
      set_item_property ('LINES.SUGGESTED_VENDOR_NAME', validate_from_list, property_true);
      set_item_property ('LINES.SUGGESTED_VENDOR_LOCATION', validate_from_list, property_true);
      --
```

Another CUSTOM.pll extension that we created provided our users the ability to print requisitions and access other custom developed functionality (software Catalog and Requisition Routing) from within the Oracle EBS Enter Requisitions form (see below Tools Menu drop down). This custom extension allowed our users to perform these tasks without having to navigate back to the main Oracle EBS purchasing navigation form.



The associated custom Tools menu item code contains the following elements:

```
--
app_special2.instantiate ('SPECIAL15', '&Print Requisition', 'POCRQERQ', TRUE, 'LINE' );
app_special2.instantiate('SPECIAL4','&Requisition Routing','APLCDISL',TRUE,'LINE');
app_special2.enable ('SPECIAL4', property_on);
```

7. Custom Oracle Forms Development

Several years ago APL had a serious need and requirement for an automated purchase requisition routing approval system. At the time, and in some cases still true today, the associated Oracle purchasing product, corresponding workflow and approvals process were too basic and maintenance heavy to be deployed at APL. We needed a system that had the following attributes:

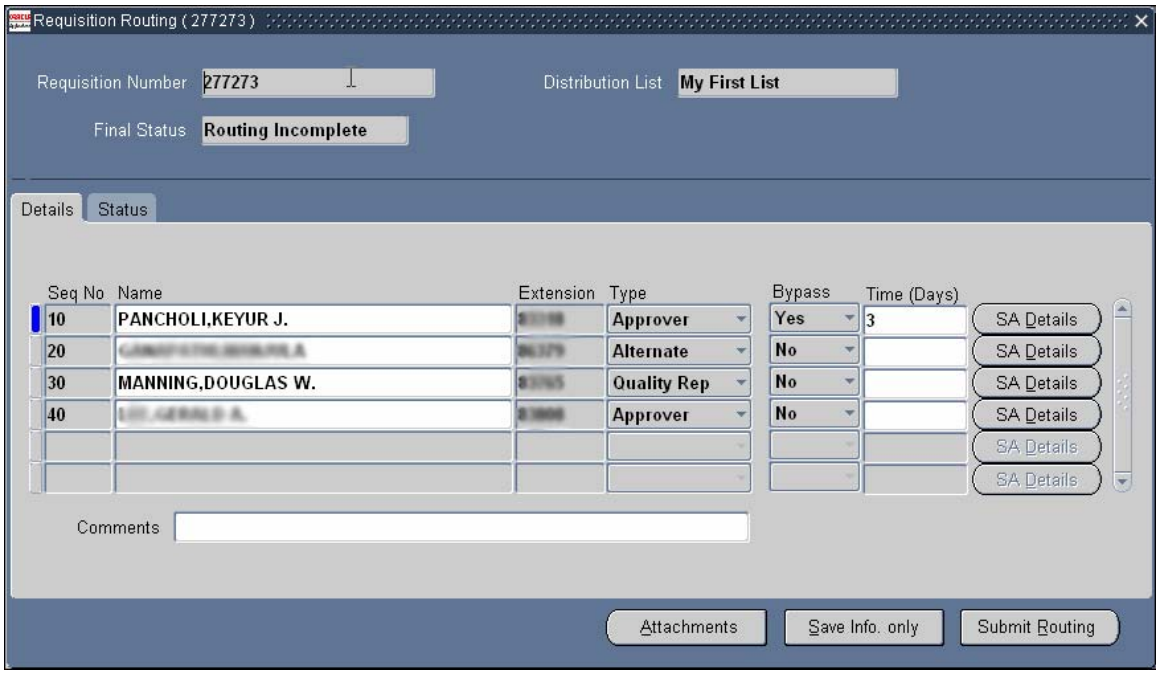
- Route requisitions through approval paths that can vary by department, type of requisition, or individual program needs
- Send supporting materials electronically, along with the requisition
- Be able to determine the status of a requisition approval at any point along the approval process
- Be able to change a routing path “on the fly”
- Cost effective to support and maintain

After exploring many options, we decided to build our own system using delivered Oracle tools (including Oracle forms and MOD PL/SQL). The following information details how APL developed and incorporated a custom designed extension process (the right way) using Oracle and html forms and the MOD PL/SQL, to provide this capability to our users. For a more comprehensive “deep-dive” into this custom process, please

reference the APL COLLABORATE06 white paper, titled “Obtaining the Seal of Approval: Automated Requisition Routing and Approval.”

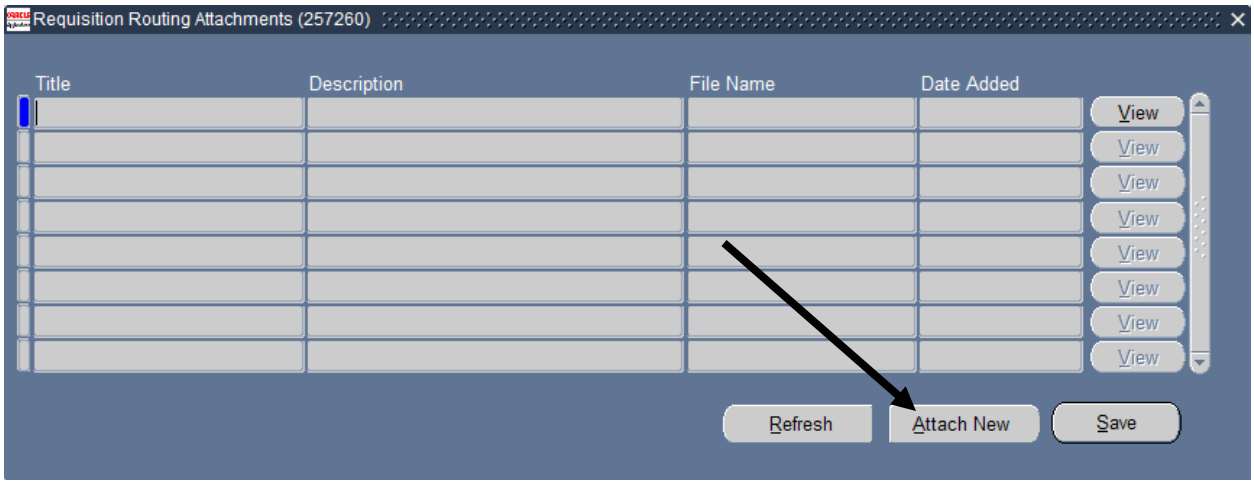
The first custom form we built (using the same look and feel as our standard Oracle EBS product) was one that could act as a template for capturing the requisition to be approved, the preparer, the requisition approvers, the type of requisition approver they were and the order of approval (see below custom APL Requisition Routing Distribution List Oracle form):

The second form we built was the purchase requisition routing process form. This form was used by requisition preparers to submit routing once the purchase requisition had been created and was also used as a gateway to our custom developed MOD PL/SQL web form, that were used to retrieve requisition related attachment documents.

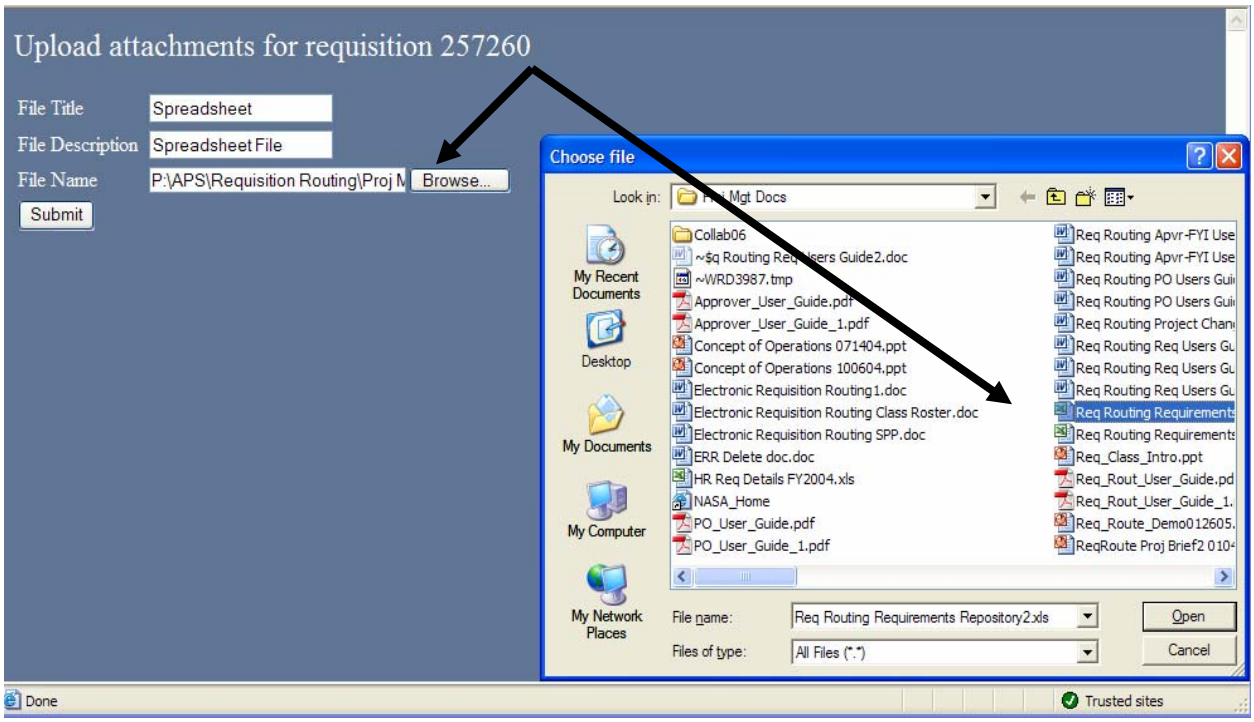


The remainder custom developed Oracle forms and MOD PL/SQL web forms were built to process and track purchase requisition attachments. The following example outlines the custom forms and associated process:

"After clicking the 'Attachment' button on the 'Requisition Routing' form, a new form will display called 'Requisition Routing Attachments' (below figure). Requisition document attachments are created and displayed on this form.

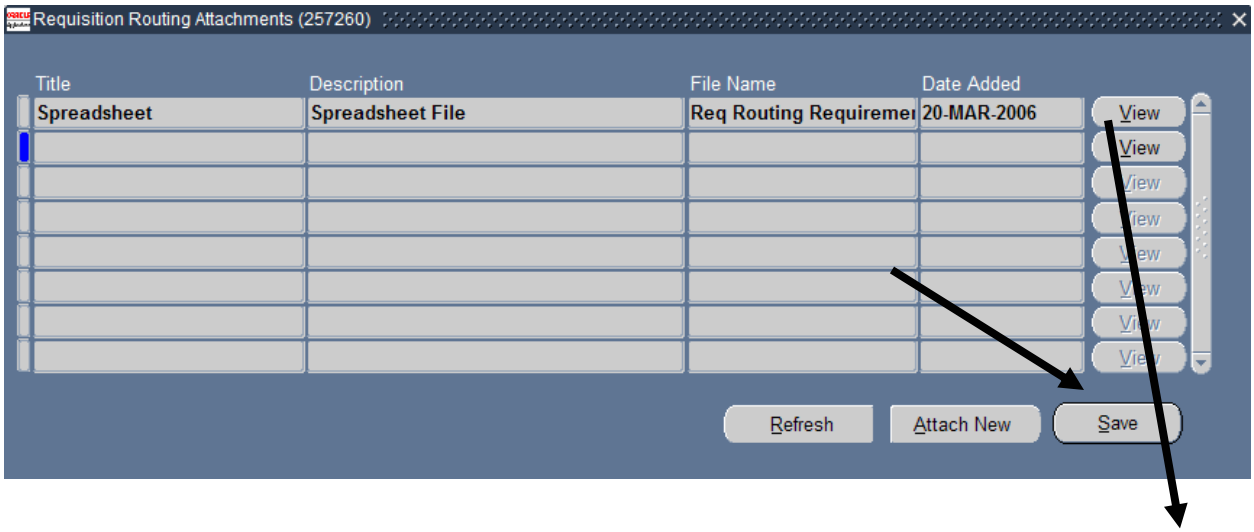


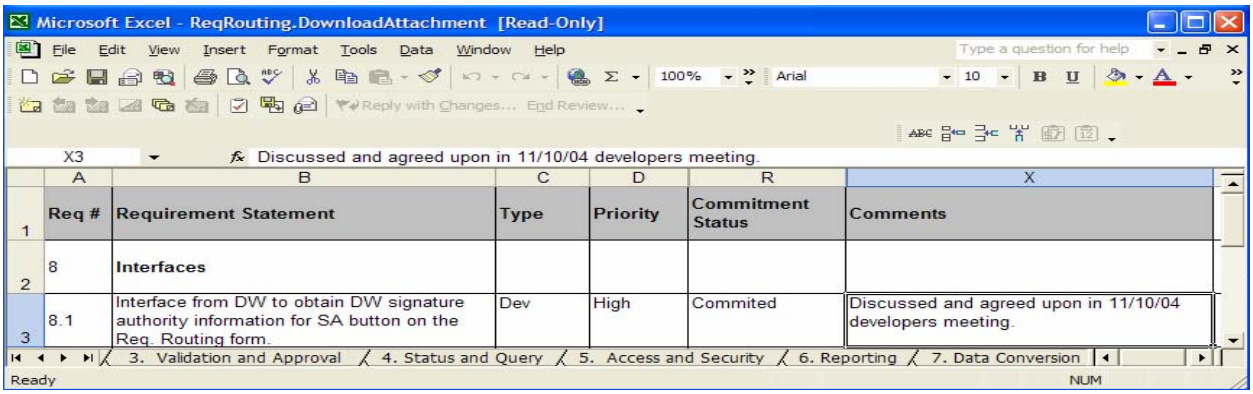
Clicking the 'Attach New' button on this form executes a PL/SQL procedure that dynamically generates an HTML page (below figure) for attachment file identification, selection and submission. The generated web page contains a 'Browse' button that allows a user to search their client or network for file objects to associate with (attach to) a specified requisition.



After entering all of the requisite fields on the 'Upload Attachments' form (previous page) and then clicking the 'Submit' button, another PL/SQL procedure executes that dynamically generates an HTML page (below figure) that informs the user if his or her document attachment upload was successful.

After successful upload, a user would return to the 'Requisition Routing Attachments' form to verify (View Button) and save (Save button) their document attachment (below figures). The title, description, file name and date when the document/file was attached to the requisition would be displayed on this form.

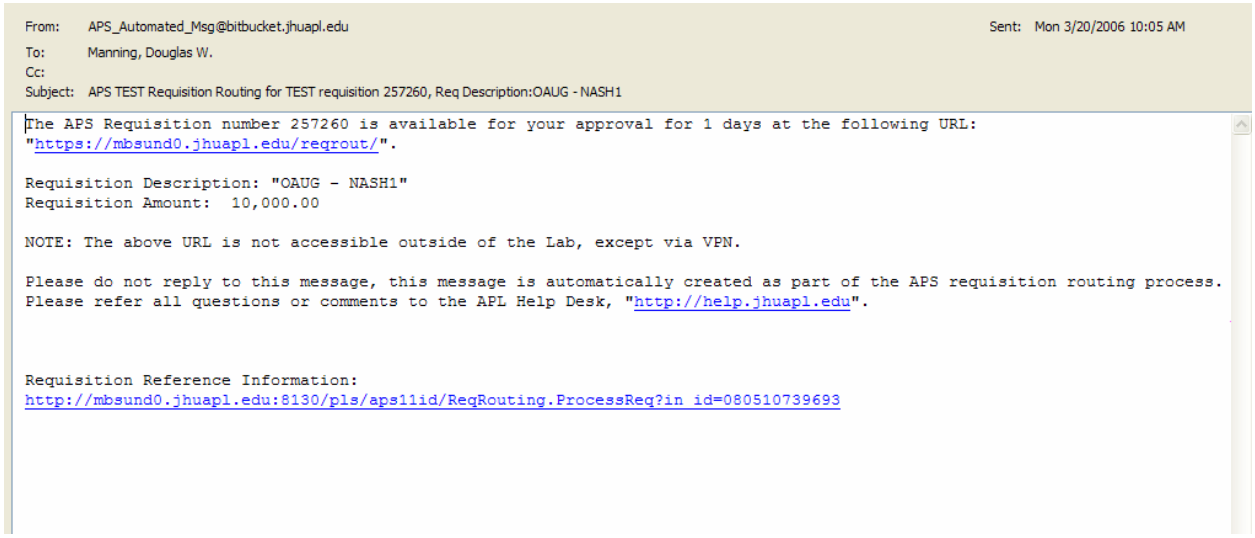




Virtually, an unlimited number of documents can be attached to a requisition using this form. The maximum document attachment file size is approximately the file size of a LONG RAW DB file type." [5]

The final forms that were built for this process involved MOD PL/SQL procedures and html web forms. These process forms alerted requisite APL employees to take action on the associated requisition, informed them of the latest status and allowed for requisition approval or rejection. The following screen shots display this functionality:

MOD PL/SQL generated email notification for Requisition Approval:



From: APS_Automated_Msg@bitbucket.jhuapl.edu Sent: Mon 3/20/2006 10:05 AM
To: Manning, Douglas W.
Cc:
Subject: APS TEST Requisition Routing for TEST requisition 257260, Req Description:OAUG - NASH1

The APS Requisition number 257260 is available for your approval for 1 days at the following URL:
"https://mbsund0.jhuapl.edu/reqrout/".

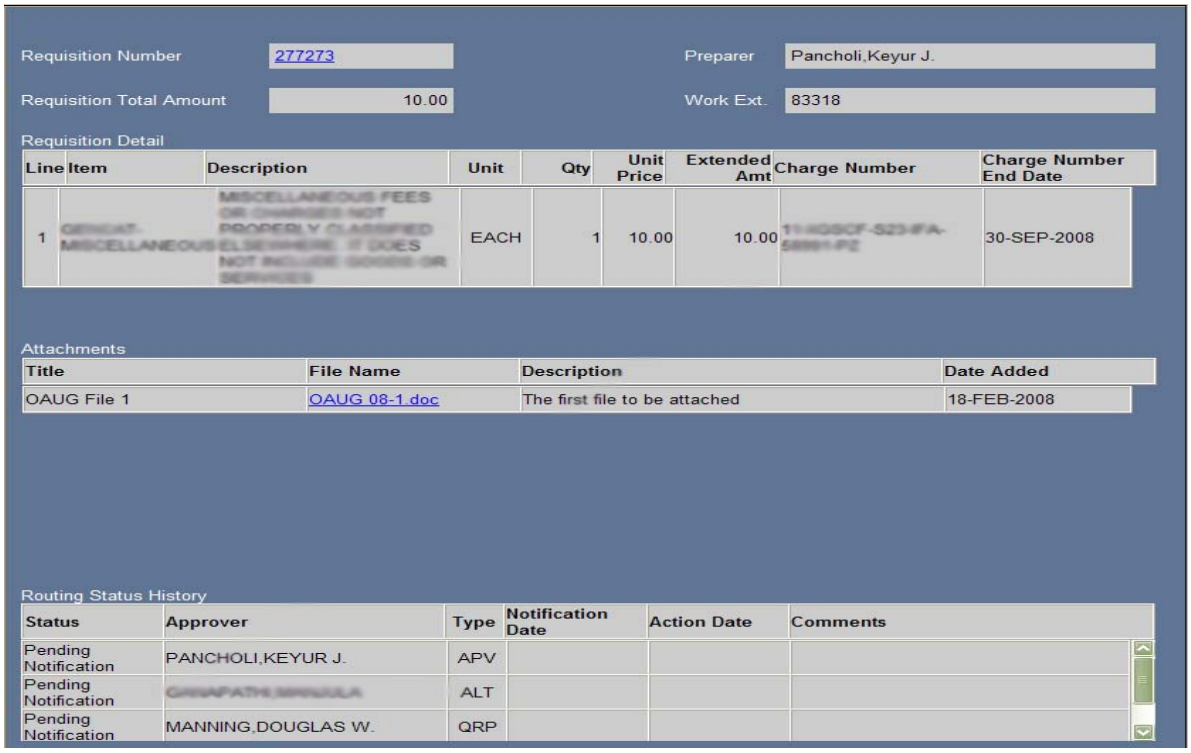
Requisition Description: "OAUG - NASH1"
Requisition Amount: 10,000.00

NOTE: The above URL is not accessible outside of the Lab, except via VPN.

Please do not reply to this message, this message is automatically created as part of the APS requisition routing process.
Please refer all questions or comments to the APL Help Desk, "http://help.jhuapl.edu".

Requisition Reference Information:
http://mbsund0.jhuapl.edu:8130/pls/aps11id/ReqRouting.ProcessReq?in_id=080510739693

APL Requisition Approval web form:



Requisition Number 277273 Preparer Pancholi,Keyur J.
Requisition Total Amount 10.00 Work Ext. 83318

Requisition Detail

Line	Item	Description	Unit	Qty	Unit Price	Extended Amt	Charge Number	Charge Number End Date
1	[redacted]	[redacted]	EACH	1	10.00	10.00	[redacted]	30-SEP-2008

Attachments

Title	File Name	Description	Date Added
OAUG File 1	OAUG 08-1.doc	The first file to be attached	18-FEB-2008

Routing Status History

Status	Approver	Type	Notification Date	Action Date	Comments
Pending Notification	PANCHOLI,KEYUR J.	APV			
Pending Notification	[redacted]	ALT			
Pending Notification	MANNING,DOUGLAS W.	QRP			

In the years since our development and deployment of this very successful custom augmentation of the Oracle EBS, Oracle has developed a similar process called the Approvals Management Engine (AME). As we pursue

our deployment of Oracle's latest version of the EBS (R12), we intend to comprehensively evaluate this process to see if this can be used to replace our very successful custom extension.

8. Oracle JDeveloper

Several years ago there was a very strong need for JHU-APL to provide a system and process that would allow the organization to easily and comprehensively rate and report upon suppliers that had provided goods and/or services to our organization. At the time our current version, and Oracle's latest version, of the EBS did not provide the requisite functionality to perform this activity in the desired fashion.

Due to the importance of this requirement to our organization and based on the insufficient capability of the Oracle EBS (at the time) to provide the level of product functionality that we desired, JHU-APL decided to design and develop an Oracle EBS "Supplier Rating" system using Oracle's latest web development technology, Oracle JDeveloper.

The following information details how JHU-APL developed core Oracle JDeveloper web forms and processes and incorporated this custom designed extension (the right way) in order to provide a processing solution to our organization within our existing Oracle EBS product:

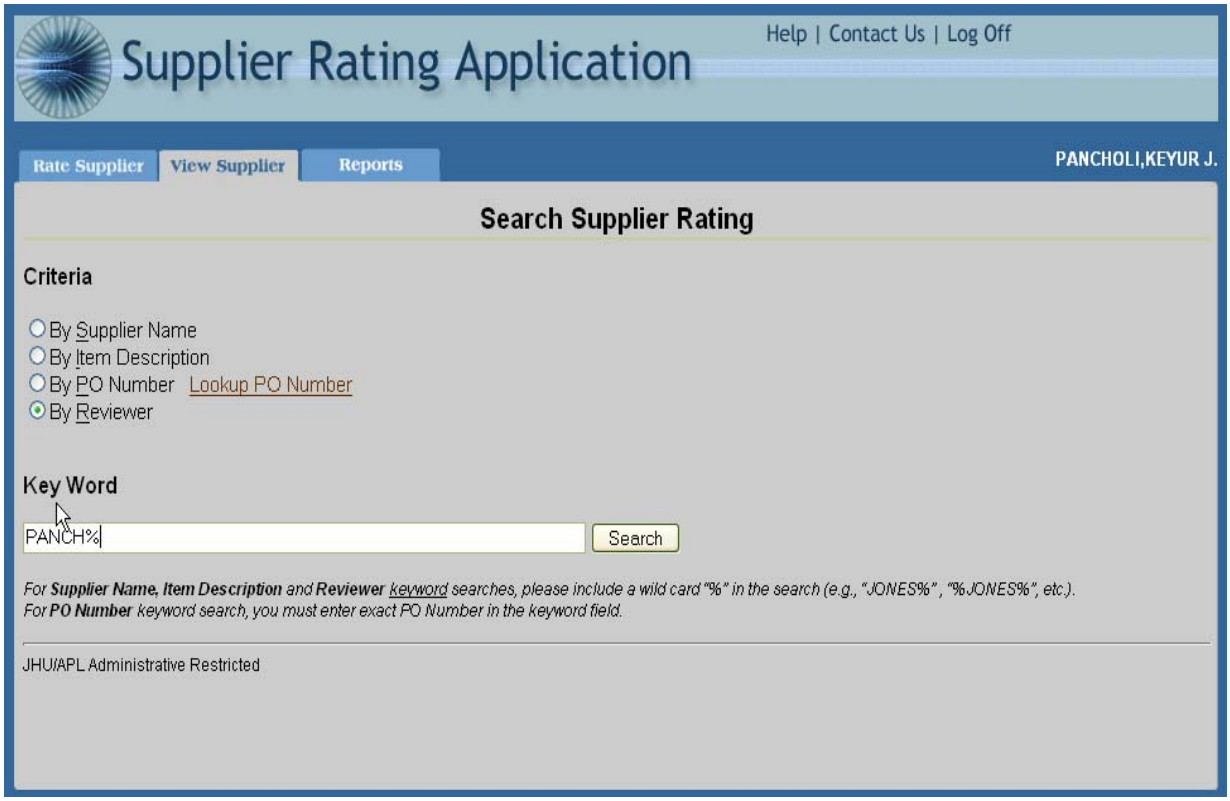
The first part of any supplier scorecard system is to have functionality that allows a customer to rate a specific supplier. The APL Supplier Rating product performed this by designing and deploying the below Oracle JDeveloper "Rate Supplier" web form:

This web form allowed our customers (APL employees) to view the specific supplier, PO, PO description and type of goods and/or services that were purchased and that they were obligated to rate. Simply clicking on the PO link in the above form provided a comprehensive supplier rating score sheet (see below form) that would be filled out by the corresponding customer:

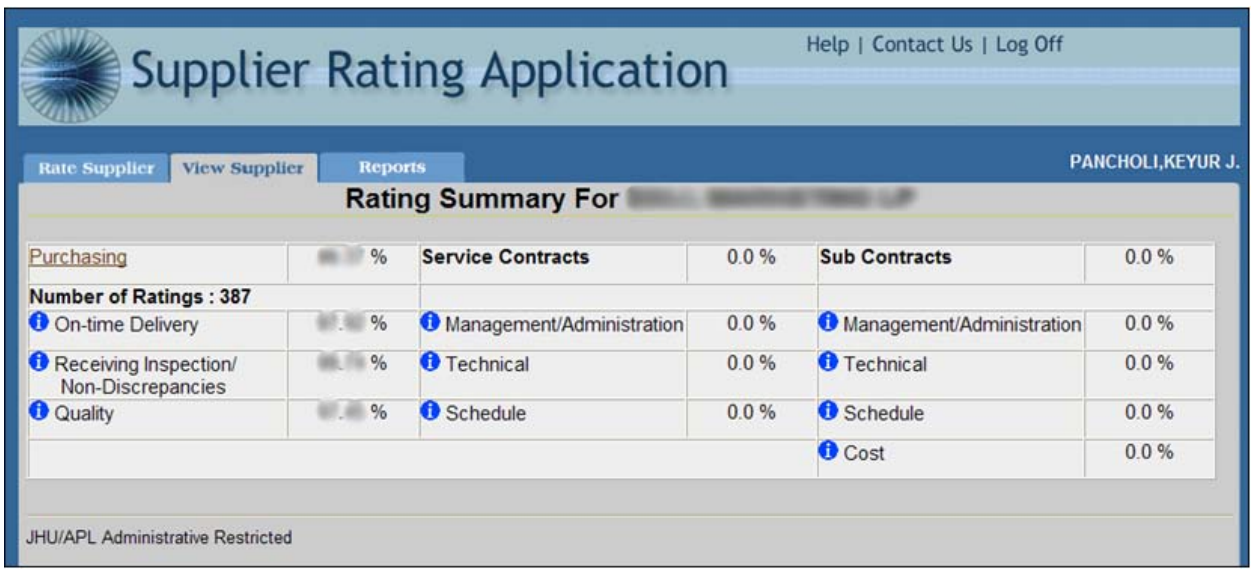
A sample of JSP code that comprises the above 'Rate Supplier' information is presented in the below text:

```
<%@ page content Type="text/html;charset=windows-1252"%>
<%@ page errorPage="/ErrorPage.jsp?page=Pending Rating List" %>
<%@ page import="model.RatingInfoList"%>
<%@ page import="model.RatingInfo"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jstl/core"%>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=windows-1252"/>
    <title>Rating List</title>
    <link href="css/Rating.css" rel="stylesheet" media="screen"/>
    <script>
      function ratingInfo(pIndex){
        document.ratingListForm.ratingInfo.value = pIndex;
        document.ratingListForm.submit();
      }
      function lookup(){
        window.open('http://mbsunp0.jhuapl.edu:8180/pls/aps11ip_apps/poc_wq_intf_pkg.po_search' ,
                'POLookup','toolbar=yes,scrollbars=yes,status=no,width=800,height=600,resizable=yes');
      }
```

The second part of this custom developed Oracle JDeveloper system involved the functionality to "View" historical APL supplier scores. The APL Supplier Rating product provided this feature through the following web form:



Any valid APL employee, using the above web form can view how well a specific supplier is performing. Search criteria can be by supplier name, by item provided, by PO number and by APL supplier reviewer. A user simply selects their criteria and then enters that criterion into the 'Key Word' text area. Once the 'Submit' button is entered the process will display the associated rating for that APL supplier, based on the criteria entered (see below form):



Depending on the type of items that a specific supplier supplies (goods, service contracts or sub contracts), the form would show the specific scores for that supplier by item and by rating criteria (see above form). Some sample JSP code used to present the above “View Supplier” web form data would include the following:

```
<h1 align="center">Search Supplier Rating</h1>
   <strong><font color="Red">
     <c:out value="${requestScope.errorMsg}"/>
   </font></strong>
   <c:set var="supplierChecked" value=""/>
   <c:set var="itemChecked" value=""/>
   <c:set var="poChecked" value=""/>
   <c:set var="reviewerChecked" value=""/>
   <c:choose>
     <c:when test="${param.searchType == '1'}">
       <c:set var="supplierChecked" value="checked"/>
     </c:when>
     <c:when test="${param.searchType == '2'}">
       <c:set var="itemChecked" value="checked"/>
     </c:when>
     <c:when test="${param.searchType == '3'}">
       <c:set var="poChecked" value="checked"/>
     </c:when>
     <c:when test="${param.searchType == '4'}">
       <c:set var="reviewerChecked" value="checked"/>
     </c:when>
     <c:otherwise>
       <c:set var="supplierChecked" value="checked"/>
     </c:otherwise>
</c:choose>
```

The third and last core component area of the APL Supplier Rating product is reporting. The below forms display how APL supplier rating data can be accessed via this custom Oracle JDeveloper process:

Again, any valid APL employee account can access and run these various APL Supplier rating reports (from the above form) by clicking on the “Reports” tab, selecting the associated report to run and then clicking the ‘Run Report’ button. Sample web report data are displayed below:

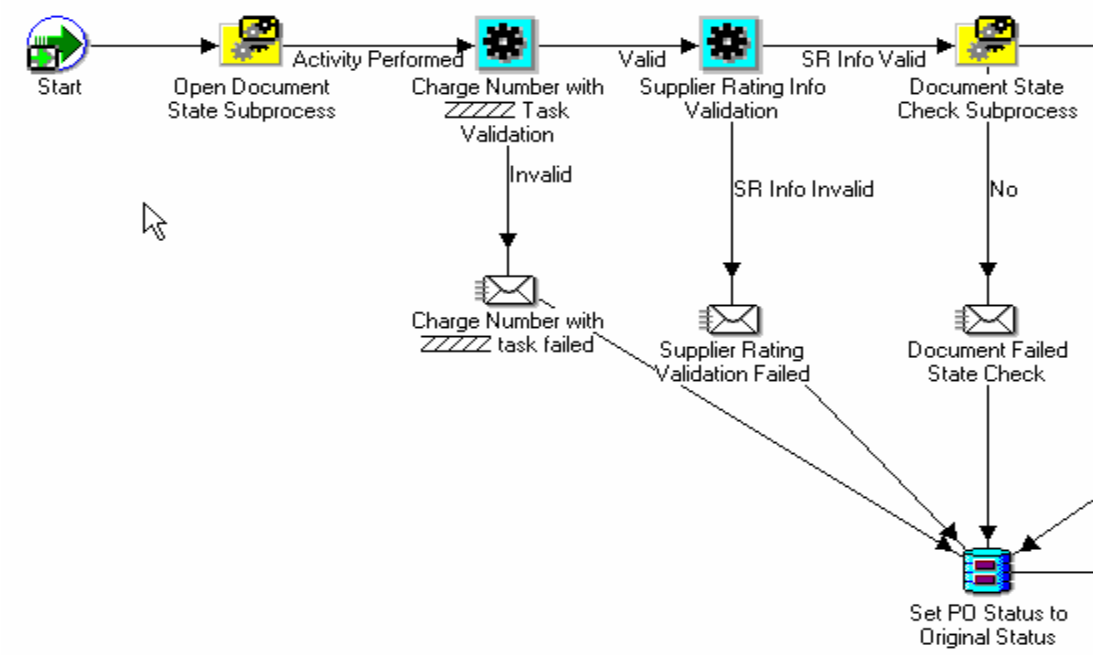
Some sample JSP code used to present the above reporting web form data would include the following:

```
checked="checked"/>Goods
    <br/>
          
    <input type="radio" name="ratingType" value="S"/>Service
    <br/>
          
    <input type="radio" name="ratingType" value="C"/>SubContract
    <br/>
          
    <input type="radio" name="ratingType" value="-1"/>Overall
    <br/><br/>
    <input type="radio" name="reportNumber" value="4"/>
    Acceptable Suppliers
    <br/>
          
    <input type="radio" name="ratingType1" value="G"
```

For a more comprehensive “deep-dive” into this custom process, please refer to APL’s Oracle OpenWorld 06 white paper titled, “Supplier Scorecard: Rating Your Buying Experience.”

9. Oracle Workflow

APL has also designed and deployed several custom extensions to the delivered Oracle EBS workflow transaction process. The below example details one of our custom extensions to the standard Oracle EBS PO approval workflow, where an APL "buyer" would receive a notification if a PO validation approval failed due to an incorrect entry of a charge number distribution task:



10. Certified 3rd Party Oracle Bolt-On

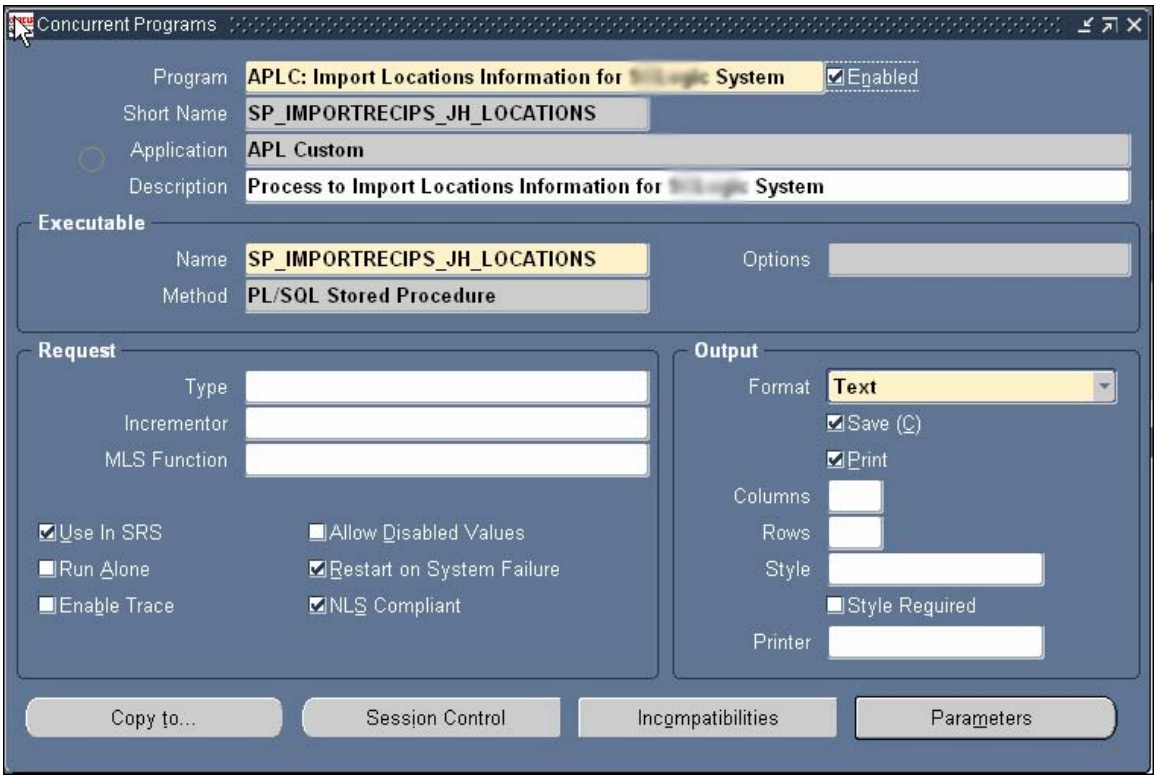
A couple of years ago our organization's receiving department came to us with several issues that they hoped we could help them resolve. In a nut shell, they were in desperate need of a system/process that would work with our currently installed Oracle EBS receiving product, which would allow them to track (real-time) parcels and packages that were to be delivered within the JHU-APL campus.

After several meetings with our receiving department, our research and analysis work with our current Oracle EBS product version capabilities, and after meeting with Oracle Corporation many times where they researched our issues and requirements to see if a current Oracle EBS product portfolio solution could cost effectively meet our needs (which did not pan out), we decided to look at a certified 3rd party bolt-on custom extension solution.

The following information details how JHU-APL incorporated a certified 3rd party product bolt-on (the right way) in order to custom extend and augment our existing Oracle EBS product:

APL's Oracle EBS "Receiving product" serves as a data source for a 3rd party bolt-on tool that captures and tracks APL in-campus parcel and package delivery information . This 3rd party software product receives data related to active APL employees and their current office delivery locations. The 3rd party software receives this data via a daily automated interface. The interface runs as a concurrent process in Oracle EBS and then pushes the data to the 3rd party product.

Here is the example of the Oracle EBS concurrent process definition for the 3rd party Parcel Tracking software product:

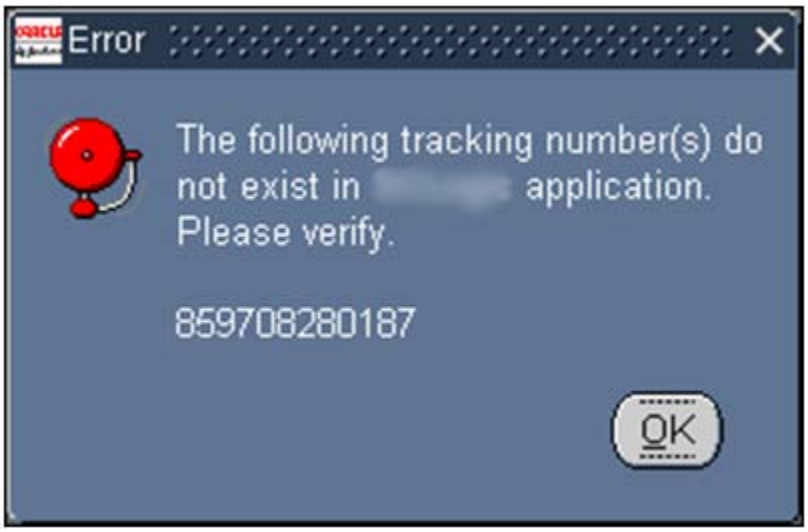


Here is the sample code for the same procedure.

```
CREATE OR REPLACE PROCEDURE INTRASYSADMIN.SP_IMPORTRECIPS_JH_PEOPLE(ERRBUF  OUT
VARCHAR2,
                                    RETCODE OUT VARCHAR2) AS

  STOO_SELCNT INTEGER;
  STOO_ERROR  INTEGER;
  STOO_ROWCNT INTEGER;
  STOO_ERRMSG VARCHAR2(255);

  --
  -- 1) Pulls recipient information from recipients (view) into a temp table (RecipImportTable)
  -- 2) Inserts any records from RecipImportTable into the Recipient table that does not already exist
  -- 3) Updates any records in the Recipient table that have changed
  -- 4) Tags records as Deleted (Status='Deleted') that are in the Recipient Table but not in the RecipImportTable
  --
  --
BEGIN
  APPS_FND.FND_FILE.PUT_LINE(APPS_FND.FND_FILE.OUTPUT,
                             '>>> Process Started : ' || TO_CHAR(SYSDATE,
'DD-MON-YYYY HH:MI:SS'));
    -- Delete records from temp import table.
    --

.. .. ..
```

```
.. .. ..
.. .. ..
 --
 COMMIT;
 --
 APPS_FND.FND_FILE.PUT_LINE(APPS_FND.FND_FILE.OUTPUT,
                  '<<< Process Ended : ' ||
                  TO_CHAR(SYSDATE, 'DD-MON-YYYY HH:MI:SS'));
EXCEPTION
 WHEN OTHERS THEN
  STOO_ROWCNT := 0;
  STOO_SELCNT := 0;
  STOO_ERROR  := SQLCODE;
  STOO_ERRMSG := SQLERRM;
  RAISE_APPLICATION_ERROR(SQLCODE, SQLERRM, TRUE);
  ROLLBACK;

END;

 -- Procedure
```

When a package is received into the Oracle EBS purchasing receiving process, a receiving clerk enters the package carrier's tracking number into the "Bill of Lading" field. As a business rule, it is assumed that the package has already been received into the 3rd party parcel tracking software from the delivery truck, and if that is not the case, the receiving clerk using the Oracle EBS will get the following error:



This way the tracking number data is checked in real-time for the existence in the tool.

Once a user successfully saves the receiving transaction in Oracle EBS, it then transfers the final recipient information back to the tool. At that point, destination data are updated from "*Unknown*" to the intended destination information.

3rd Party Package Tracking Software Screen-Shot

Date/Time	Status	Destination	CurrentID	Carrier	Parent ID	Note	Comments	More Comments
2/15/2008 11:27:32 AM	Delivered	JR.,GEORGE B. ( )	14-125 (14-125)	summeij1		N/A	N/A	N/A
2/15/2008 8:32:54 AM	Out for Delivery	JR.,GEORGE B. ( )	DELIVERY1 (DELIVERY1)	summeij1		N/A	N/A	N/A
2/14/2008 4:58:39 PM	Building 31 Shelf	JR.,GEORGE B. ( )	ZONE1 (ZONE1)	grandss1		N/A	N/A	N/A
2/14/2008 3:05:43 PM	On Receivers Desk	JR.,GEORGE B. ( )	CHECKER4 (CHECKER4)	OracleFinancials		N/A	N/A	N/A
2/14/2008 9:28:20 AM	On Receivers Desk	*Unknown* (unknown)	CHECKER4 (CHECKER4)	grandss1		N/A	N/A	N/A
2/13/2008 1:19:02 PM	Receive	*Unknown* (unknown)	*Unknown* (Unknown)	gaithlp1		N/A	N/A	N/A

Oracle EBS Custom Extensions (R12 and Beyond)

Developing and deploying Oracle EBS custom extensions in R12 in the application forms (Oracle forms and web forms) area will probably center on two major technology tracks:

- Oracle Forms (OF) based application.
- Oracle Application Framework (OAF)

The CUSTOM.pll library will still be available in R12 for building custom extension overrides to R12 Oracle forms. Some new functionality that was added to this process structure includes the following:

WHEN-RESPONSIBILITY-CHANGED
This fires immediately when user selects "File→ Switch Responsibility". Note that you cannot get information about the responsibility user leaves.

WHEN-LOGON-CHANGED
This fires immediately when user selects "File→ Log On as a Different User". Here also, you cannot get information about the previous user.

In addition, CUSTOM.STYLE, is also available in many product areas. With this added product feature, a developer can choose to have their code execute before, after or in place of the code provided in Oracle Applications. This functionality does not affect all generic form events that are available in the custom library and some product-specific events may not support all execution styles.

If you have developed custom extension functionality in previous versions using the MOD PL/SQL structure, you will need to rewrite/re-establish this functionality using Oracle APEX and the 10gR2 Embedded SQL Gateway or recoding using OAF.

Conclusion

How do organizations today and in the future world of Oracle EBS software solutions maximize the cost, efficiency and technology benefits that the Oracle EBS product provides, without losing the ability to capture, transact, retrieve and report on unique and important data that are essential and vital to their organizations?

They do it first by rigorously contemplating, designing, developing and implementing solutions within their organizations that are business process driven (not involving the modification of COTS, Oracle EBS product software), and if that is impossible and cannot be done, they develop Oracle EBS custom extensions the right way.

Notes

[1] Brownsword, Lisa. “The Good News About COTS.” News @ Software Engineering Institute, pg 2 (1Q03).

[2] U.S. Department of Defense. “Modifying Commercial Off-the-Shelf (COTS) Software.” Defense Acquisition Guide, section 7.10.4 (December 16th, 2004)

[3] Oracle Corporation. “Customization and Development for E-Business Suite.” Oracle Technology Network (June 2006)

[4] U.S. Department of Defense. “Modifying Commercial Off-the-Shelf (COTS) Software.” Defense Acquisition Guide, section 7.10.4 (December 16th, 2004)

[5] Manning, Douglas; Payne, Raymond; Tamer, John The Johns Hopkins University Applied Physics Laboratory. “Obtaining the Seal of Approval: Automated Requisition Routing.”, OAUG COLLABORATE06, pgs. 15-16 (April 2006)